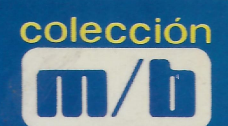
SERIE
GUIA PRACTICA



ZX microdrive

andrew pennell

- programas
- código máquina
- red de área local

LAS POSIBILIDADES "OCULTAS" DEL MICRODRIVE AL DESCUBIERTO



EL ZX Spectrum CONVERTIDO EN UNA POTENTE MÁQUINA CON LOS MICRODRIVES

zx microdrive

ANDREW PENNELL

zx microdrive

EDICIONES TECNICAS REDE, S.A.
Apartado 35.400
BARCELONA

Título original: MASTER YOUR ZX MICRODRIVE

1.ª Edición en inglés en 1983
Publicada por Sunshine Books (de Scot Press Ltd.)
12/13 Little Newport Street
Londres - WCR 3LD (Gran Bretaña)

Traducción: Ramón Sangüesa Solé

ISBN: 84-247-0197-6

Impreso en España Printed in Spain

Dep. Legal: B. 29234/1984 —REDEPRINT— Barcelona

Sumario

INTRODUCCION

Mi objetivo al escribir este libro ha sido enseñar la forma de utilizar mejor el Interface 1 de Sinclair, haciendo especial referencia a los Microdrives ZX. El Interface amplía en gran medida la utilidad del Spectrum y los Microdrives ofrecen un almacenamiento de masa de acceso rápido a un precio muy bajo.

En este libro se explican claramente todas las características típicas del Interface y los Microdrives pero, además, se tratan toda una serie de posibilidades «ocultas» que convierten a estos dispositivos en unidades mucho más potentes y útiles. Se incluyen diversas rutinas en código máquina destinadas a incrementar las posibilidades del Interface. La presentación de estas rutinas se ha hecho de forma que cualquiera pueda utilizarlas aunque no posea conocimiento alguno acerca de la programación en código máquina. Para aquellos lectores que sí conozcan el tema he preparado una sección que trata sobre las rutinas de la ROM del Interface y de cómo emplearlas. En esa misma sección se pueden encontrar los listados comentados de mis propias rutinas.

Los programas en código máquina se dan siempre en forma de una serie de números almacenados en una o varias sentencias DATA a las que acompaña un bucle FOR-NEXT encargado de colocar esos números en el lugar adecuado mediante instrucciones POKE. Siempre que ha sido posible, el código máquina se ha hecho independiente de la posición. Esto quiere decir que, cualquiera que sea la posición de memoria en que se cargue, el programa en código máquina funcionará perfectamente. El mejor lugar para colocar programas en código máquina se encuentra por encima de la posición de memoria RAMTOP (ver página 179 del Manual del Spectrum). Esta posición puede desplazarse hacia abajo mediante la sentencia CLEAR. Otros lugares donde se pueden cargar programas escritos en código

máquina son el área destinada a almacenar gráficos definidos por el usuario (168 bytes) y en la memoria intermedia («buffer») de la impresora (256 bytes). Cuando se utilice el Interface deberá evitarse, en lo posible, el empleo de instrucciones REM como lugares de almacenamiento de código máquina. Hay razones técnicas que impiden recurrir a ellas para estas misiones. Con el fin de facilitar al máximo su utilización acompaño cada rutina de este libro con dos posiciones recomendadas (una para las máquinas que utilizan 16 K y otra para las que emplean 48 K). Estas dos posiciones permiten que todas las rutinas se encuentren presentes simultáneamente. Si se utilizan, hay que desplazar primeramente la posición RAMTOP. Para todas las rutinas se obtendrá espacio suficiente haciendo CLEAR 32009 (para máquinas con 16 K) o CLEAR 64779 (para 48 K). Como en alguna de las rutinas de mayor extensión aparecen gran cantidad de números, la posibilidad de que haya errores en el momento de transcribirlas es mayor. Por este motivo todas las rutinas van formando una suma de control («checksum») a medida que realizan instrucciones POKE. Si la suma total obtenida no es correcta, la rutina se detendrá dando el mensaje correspondiente. Entonces habrá que comprobar los datos entrados y corregirlos si fuera necesario.

Otras secciones del libro están dedicadas a la seguridad de los programas y a cómo ampliar el BASIC del Spectrum mediante la adición de nuevas instrucciones. En los apéndices se ha reunido información de tipo más técnico así como explicaciones acerca de los errores típicos que pueden producirse al emplear el Interface.

La preparación de este libro es en sí misma una buena muestra del potencial del Spectrum como ordenador «serio»: he utilizado un Spectrum de 48 K, con un teclado profesional, un interfaz para impresora, un procesador de textos y una impresora de agujas.

Andrew Pennell
Cliftonville, Kent
Septiembre 1983

Capítulo 1

CORRIENTES Y CANALES

Antes de entrar a discutir en detalle las muchas posibilidades adicionales que el Interface ZX confiere al Spectrum, es necesario comprender perfectamente los conceptos de «corriente» y «canal» puesto que resultan fundamentales para utilizar con eficacia tanto los Microdrives como el RS232 y las posibilidades de enlace a una red.

Un **canal** es aquella parte del sistema de un ordenador a la que se pueden enviar o de la que se pueden recibir datos. Entre otros ejemplos de canales se pueden citar: el teclado —para entrada de datos— y la pantalla —para salida de datos—. Las rutas o caminos que deben seguir los datos para llegar a los canales reciben el nombre de **corrientes.**

En el Spectrum se puede disponer, desde el BASIC, de 16 corrientes distintas. De esas 16, hay cuatro asignadas inicialmente a unos canales determinados según se muestra en la siguiente tabla:

Corriente N.°	Identificador del canal	Utilización de salida	Utilización de entrada
# 0	«K»......	a la parte inferior de la pantalla	desde el teclado
# 1	«K»......	a la parte inferior de la pantalla	desde el teclado
# 2	«S»......	a la parte superior de la pantalla	no puede ser entrada
# 3	«P»......	a la impresora ZX	no puede ser entrada

Las corrientes 0 y 1 son idénticas y están conectadas al canal «K». Este canal envía los caracteres a la parte inferior de la pantalla (donde aparecen los mensajes de error y las operaciones INPUTs) y también los recibe desde el teclado. La corriente 2 es el canal «S», encargado de imprimir los caracteres en la parte superior de la pantalla. La corriente 3 es el canal «P», que envía los caracteres a la impresora ZX. El sistema del Spectrum impide que haya entradas a través de las corrientes 2 y 3. Por ejemplo, si se intenta ejecutar la sentencia INPUT # 3;a$, se provocará la aparición del mensaje de error **Invalid I/O device** (dispositivo de Entrada/Salida incorrecto).

El símbolo utilizado para designar a las corrientes es el # y puede añadirse a las instrucciones PRINT, INPUT y algunas otras. Por ejemplo, para imprimir en las líneas inferiores de la pantalla —cosa que normalmente es imposible— se puede recurrir a las corrientes 0 ó 1:

```
PRINT #0;AT 0,0;"LINEA 23";
AT 1,0;"LINEA 24";: PAUSE 0
```

Es necesario emplear una pausa con el fin de impedir que con el mensaje «OK» se destruya el mensaje que se quería imprimir en esas líneas. Al imprimir a través del canal «K» puede ocurrir que inesperadamente la parte inferior de la pantalla se desplace verticalmente, superponiéndose sobre la parte superior de la pantalla. Se puede evitar este inconveniente mediante el empleo de instrucciones AT así como terminando cualquier texto con un punto y coma.

La sentencia PRINT # 2; tiene el mismo efecto que la instrucción PRINT normal que envía caracteres al canal «S» (a la parte superior de la pantalla). Con PRINT # 3; el resultado es el mismo que con LPRINT, que imprime caracteres en la impresora ZX (canal «P»). Una ventaja al utilizar las corrientes 2 y 3 en nuestros programas es que se puede tener una variable para determinar si la salida de datos debe ser a través de la pantalla o de la impresora. El siguiente programa demuestra la utilización de esta técnica:

```
1000 INPUT "IMPRESORA (S/N)";A$
1010 LET C=2: IF A$="s" OR A$="S
" THEN  LET C=3
1020 PRINT #C;"ESTO ES UN ZX SPE
CTRUM"
```

También se pueden emplear dos corrientes distintas en una misma instrucción PRINT. Por ejemplo:

```
2000 PRINT #2;"LA PANTALLA";#3;"
LA IMPRESORA ZX"
```

Asimismo se pueden utilizar las instrucciones INPUT y LIST con corrientes. En un Spectrum que no disponga del Interface sólo está permitido hacer INPUT # 0 ó INPUT # 1, instrucciones que resultan equivalentes a la INPUT normal. Sin embargo, como veremos en posteriores capítulos, es extremadamente útil el empleo de instrucciones del tipo INPUT # cuando se posee el Interface.

La sentencia LIST # n (donde «n» designa el número de la corriente) lleva el listado de un programa en BASIC a la corriente correspondiente. La sentencia LIST # 3 es equivalente a la LLIST que saca un listado por la impresora ZX. Con LIST # 0 se obtiene un efecto muy interesante —aunque sin ninguna utilidad— en la pantalla.

También se puede emplear la sentencia INKEY # n con las corrientes aunque no tiene mucha utilidad con los canales normales. Las sentencias INKEY$ # 0 e INKEY$ # 1, dan normalmente como resultado cadenas vacías. No está permitido utilizar los números # 2 y # 3 con estas sentencias. En capítulos posteriores se explica cómo hacerlo con otros canales del Interface.

OPEN # Y CLOSE

En la configuración básica del Spectrum la utilidad del comando OPEN # queda bastante restringida. El objetivo de esta instruc-

ción es asignar un canal a una corriente. La forma general del comando es:

```
OPEN #N,F$
```

donde «n» designa el número de la corriente y f$ al identificador del canal. Obsérvese que el signo # es parte de la palabra clave OPEN # obtenida mediante la pulsación de las teclas Symbol Shift-3. El valor de «n» tiene que estar comprendido entre el 0 y el 15 (de otra forma aparecerá el error Corriente no válida **(Invalid stream)**. El identificador del canal tiene que ser una sola letra y debe indicar un canal correcto. Se pueden utilizar letras en mayúscula o en minúscula es decir: «K», «k», «S», «s», «P» o «p». Si se intenta utilizar una letra no válida, aparecerá el error Nombre de fichero no válido **(Invalid filename)**. Cuando el Interface 1 está conectado también son correctos los identificadores «M», «N», «B» y «T», así como sus correspondientes versiones en minúscula. Estos identificadores designan a los canales del Microdrive, enlace a red de ordenadores, RS232 Binario y RS232 de Texto respectivamente. Con estos canales adicionales también se puede utilizar un punto y coma como separador en las sentencias OPEN #. Es posible también emplear una coma. Algunos de estos canales necesitan otros datos adicionales. Se darán los detalles pertinentes en los capítulos correspondientes.

Por ejemplo:

```
OPEN #3;"s"
```

hará que toda la salida de la corriente 3, que normalmente va a la impresora ZX, vaya a la parte superior de la pantalla. De esta forma, con LLIST se obtendrá un listado sobre la pantalla. Resultará muy conveniente recurrir a esta sentencia cuando se quiera depurar programas que utilicen la impresora ZX o bien cuando se quiera ahorrar papel. La instrucción de efecto contrario al anterior es:

```
OPEN #2,"P"
```

que hace que toda la salida de la corriente 2, que normalmente sale por la pantalla, vaya a parar a la impresora. Si ambos coman-

dos se ejecutan simultáneamente, la salida del programa será, como mínimo, bastante confusa.

Se puede emplear OPEN # para utilizar otras corrientes. Normalmente las corrientes de la 4 a la 15 son «corrientes sin canal» y cualquier intento de emplearlas (por ejemplo, PRINT # 6; «6»;) provocará la aparición del mensaje de error Corriente no válida **(Invalid stream)**. Sin embargo, es posible «abrir» estas corrientes adicionales de la forma normal:

```
OPEN #4,"s"
PRINT #4;"HOLA!"
```

hará que aparezca la palabra «¡Hola!» en la pantalla, canal «S». Cuando se emplea el Interface, las corrientes de la 4 a la 15 se utilizan mucho para poder hacer uso de otros periféricos Sinclair.

Para anular el efecto de una sentencia OPEN # n se debe emplear el comando complementario CLOSE # n, donde «n» indica el número de la corriente. Al cerrar («Close») las corrientes 0 a 3 pasarán a estar asignadas a sus canales normales. Cuando se cierran las comprendidas entre la 4 y la 15 pasarán a no tener asignado canal alguno.

IMPORTANTE: Evítese el uso de la sentencia CLOSE # con las corrientes de la 4 a la 15 si no está conectado el Interface. Debido a un error en el BASIC, si la corriente especificada está ya cerrada pueden ocurrir cosas extrañas. Normalmente, se recibe un mensaje de error un tanto curioso pero también puede suceder que todo el ordenador se inicialice. Cuando el Interface está conectado no hay problema alguno ya que el error queda automáticamente corregido.

CLEAR # Y MOVE

Existe otra sentencia que también puede utilizarse con los canales: es la CLEAR # (obsérvese el símbolo #). Este símbolo no le da ninguna importancia especial a la sentencia CLEAR. El objeto de esta sentencia es cerrar las corrientes de la 4 a la 15 y reasignar las

corrientes de la 0 a la 3 a sus canales originales. Cuando se quiera utilizar esta sentencia con los canales adicionales del Interface habrá que poner especial cuidado.

Finalmente, una última instrucción que puede emplearse en relación a los canales es la instrucción MOVE. Puede aparecer bajo diversas formas pero el formato más general es el siguiente:

```
MOVE #a TO #b
```

Donde **a** y **b** son los números de las corrientes y TO una palabra clave. Lo que hace esta instrucción es leer un carácter de la corriente «a» e imprimirlo en la corriente «b». Se repite este proceso hasta que se satisfaga una cierta condición (esta condición depende del tipo de canal al que esté conectada la corriente «a»). Se trata de un comando muy potente cuyas diferentes aplicaciones se explicarán en los siguientes capítulos.

Es posible que una rutina en código máquina genere corrientes especiales para usos particulares. Por ejemplo, el sistema operativo de BASIC emplea internamente el canal «R» en asociación con la corriente —1 (que no puede ser utilizada por programas en BASIC) para «imprimir» caracteres en el área interna de trabajo de la memoria. También es posible modificar canales ya existentes. Por ejemplo, todos los interfaces producidos por empresas independientes de Sinclair para conectar impresoras, del tipo Centronics, modifican el canal «P» de forma que toda la salida de la corriente 3 (por ejemplo, LPRINT) vaya a una impresora externa independiente, a través del interfaz en vez de salir por la impresora ZX.

Cuando se utilizan simultáneamente varias corrientes resulta difícil saber qué canal está conectado a qué corriente y qué corrientes no están realizando función alguna. El programa siguiente genera una tabla en la que se relacionan los detalles correspondientes a cada corriente, incluidas las que van de la —3 a la —1 (que sólo son accesibles a programas en código máquina). Cuando el programa se ejecuta, imprime un resumen acerca del empleo de cada corriente indicando si se puede utilizar para entrada, salida o para ambas cosas a la vez. Una vez se ha presentado el resumen, se pide al usuario del programa que entre el número de una corriente. Una vez se ha entrado, el programa da más detalles acerca de esa co-

rriente en particular. Tal y como está ahora, el programa sólo da información acerca de los canales normales. A lo largo del libro se irán ampliando las posibilidades del programa.

Catálogo de corrientes

```
1000 DEF FN p(p)=PEEK p+256*PEEK
 (p+1)
1005 CLS : PRINT TAB 6; INVERSE
1;"UTILIZACION CORRIENTES"
1010 PRINT "CORRTE.";TAB 8;"ENT/
SAL";TAB 16;"NOM.";TAB 23;"EMPLEO"
1020 FOR s=-3 TO 15
1025 PRINT TAB 2;s;
1030 LET d=FN p(s*2+23566+8)
1040 IF d=0 THEN  PRINT TAB 20;
INVERSE 1;"LIBRE": GO TO 1200
1050 LET d=d+FN p(23631)-1
1060 IF FN p(d+2)<>5572 THEN  PR
INT TAB 8;"ENT";
1070 PRINT TAB 11;"/";
1080 IF FN p(d)<>5572 THEN  PRIN
T TAB 12;"SAL";
1090 LET f$=CHR$ PEEK (d+4)
1100 PRINT TAB 16;"""";f$;"""";T
AB 20;
1110 IF f$="K" THEN  PRINT "PANT
ALLA INF": GO TO 1200
1120 IF f$="S" THEN  PRINT "PANT
ALLA SUP": GO TO 1200
1130 IF f$="P" THEN  PRINT "IMPR
ESORA": GO TO 1200
1140 IF f$="M" THEN  PRINT "MICR
ODRIVE": GO TO 1200
1150 IF f$="N" THEN  PRINT "ENLA
CE RED": GO TO 1200
1170 IF f$="T" THEN  PRINT "RS23
2": GO TO 1200
1180 IF f$="R" THEN  PRINT "AREA
 TRABAJO": GO TO 1200
```

```
1190 PRINT  FLASH 1;"SIN ESPECIF
ICAR"
1200 NEXT s
1210 PRINT AT 0,0;
1500 INPUT "ENTRAR NUMERO DE COR
RIENTE O     ENTER PARA TERMINAR
?"; LINE S$
1505 IF S$="" THEN  STOP
1510 CLS
1515 LET S=VAL S$
1520 PRINT TAB 6; INVERSE 1;"NUM
ERO DE CORRIENTE ";S''
1530 LET D=FN P(S*2+23566+8)
1540 IF D=0 THEN  PRINT "CORRIEN
TE CERRADA": GO TO 1500
1550 LET D=D+FN P(23631)-1
1560 LET F$=CHR$ PEEK (D+4)
1570 PRINT "IDENTIFICADOR DE CAN
AL:";F$
1580 REM COMPROBAR CADA TIPO
1600 IF F$="K" THEN  GO TO 2000
1610 IF F$="S" THEN  GO TO 2020
1620 IF F$="P" THEN  GO TO 2040
1660 IF F$="R" THEN  GO TO 2050
1670 PRINT  FLASH 1;"IDENTIFICAD
OR DESCONOCIDO"
1700 GO TO 1500
1800 PRINT "RUTINA DE SALIDA
";FN P(D)
1810 PRINT "RUTINA DE ENTRADA
 :";FN P(D+2)
1820 IF FN P(D)=5572 THEN  PRINT
  FLASH 1;"SOLO ENTRADA"
1830 IF FN P(D+2)=5572 THEN  PRI
NT  FLASH 1;"SOLO SALIDA"
1870 RETURN
1999 REM CANAL K
2000 PRINT  INVERSE 1;"PANTALLA
INFERIOR/TECLADO"
2010 GO TO 2060
2019 REM CANAL S
```

```
2020 PRINT  INVERSE 1;"PANTALLA
SUPERIOR"
2030 GO TO 2060
2039 REM CANAL P
2040 PRINT  INVERSE 1;"IMPRESORA ZX"
2045 GO TO 2060
2049 REM CANAL R
2050 PRINT  INVERSE 1;"AREA DE T
RABAJO"
2060 GO SUB 1800
2070 GO TO 1500
```

El programa trabaja aplicando la operación PEEK a varias posiciones de memoria (lea la siguiente explicación sólo si realmente le interesa).

El bucle FOR-NEXT **s** (línea 1020) trata sucesivamente a cada una de las corrientes. La variable **d** representa el desplazamiento en el área STRMS (1030), para la corriente seleccionada. Este desplazamiento será cero si la corriente está cerrada (1040). Este desplazamiento se suma a la variable del sistema CHANS (1050). La variable «d» apunta a una serie de bytes del área de memoria CHANS. Los canales normales (es decir, K, S, P y R) ocupan en el área CHANS un mínimo de cinco bytes cada uno lo que supone un total de 20 bytes. Los dos primeros bytes de cada bloque de cinco apuntan a la rutina de salida de un carácter. Los dos segundos apuntan a la rutina de entrada de un solo carácter. El quinto byte es el identificador del canal expresado en mayúsculas o minúsculas. La posición 5572 guarda una instrucción de salto que lleva a la rutina de error correspondiente a Dispositivo de Entrada/Salida no válido **(Invalid I/O device)**. Para comprobarlo haga RANDOMIZE USR 5572. La comprobación correspondiente se realiza en las líneas 1060 y 1080. El identificador del canal se imprime en la línea 1100 y en las líneas 1110-1180 se comprueba que designa a alguno de los canales conocidos; luego se imprime el nombre del dispositivo correspondiente. Una vez impreso el resumen, el usuario puede entrar el número de una corriente (1500) y recibir más detalles. Para los canales que hemos tratado hasta ahora esta información corresponde únicamente a las posiciones de entrada y salida (para ello se emplea la subrutina de la línea 1800). Más adelante ampliaremos este aspecto.

```
              UTILIZACION CORRIENTES
      CORRTE. ENT/SAL NOM.   EMPLEO
        -3    ENT/SAL "K" PANTALLA INF
        -2       /SAL "S" PANTALLA SUP
        -1       /SAL "R" AREA TRABAJO
        0     ENT/SAL "K" PANTALLA INF
        1     ENT/SAL "K" PANTALLA INF
        2        /SAL "S" PANTALLA SUP
        3        /SAL "P" IMPRESORA
        4     ENT/SAL "T" RS232
        5                 LIBRE
        6     ENT/SAL "N" ENLACE RED
        7                 LIBRE
        8                 LIBRE
        9                 LIBRE
        10    ENT/SAL "M" MICRODRIVE
        11                LIBRE
        12                LIBRE
        13                LIBRE
        14                LIBRE
        15
FIGURA 1. LISTADO CORRIENTES DE SALIDA
```

La **Figura 1** muestra la salida típica que se obtiene al ejecutar este programa.

Programa «corriente 14-z$»

El programa siguiente hace que la corriente 14 sea capaz de «imprimir» caracteres en una variable de BASIC de tipo cadena. Al llamar a la rutina mediante la sentencia GO SUB 8000 la variable **st** tiene que contener la dirección de la posición de memoria a la que debe dirigirse la subrutina. La rutina ocupa 101 bytes y se recomiendan para ella las posiciones de memoria 65260 (48 K) y 32490 (16 K). Una vez haya sido llamada la rutina, toda la salida de la corriente 14 se irá añadiendo al final de la variable de tipo cadena **z$** (variable que no debe estar dimensionada). Si **z$** no existe o está dimensiona-

da, se producirá un error del tipo **Variable not found** (Variable no encontrada). Es posible realizar las sentencias PRINT # 14; y LIST # 14 y, lo que quizás resulte más útil, puede utilizarse para catalogar un Microdrive dentro de la cadena. Sin embargo, existe una restricción: las sentencias que impriman cadenas directamente no funcionarán de forma correcta. Por ejemplo,

```
PRINT #14;"CADENA"
```

funcionará incorrectamente añadiendo al final de **z$** la cadena «ssssss». Para corregir este defecto se pueden emplear dos métodos: o bien se imprime cada carácter individualmente (por ejemplo, PRINT # 14; "c"; "a"; "d"; "e"; "n"; "a") o bien se emplea otra variable de tipo cadena (LET a$ = "cadena": PRINT # 14; a$). Para programadores expertos diré que la razón es que la rutina desplaza hacia arriba secciones de memoria a fin de disponer de espacio en el que insertar los caracteres pero también desplaza en el mismo sentido el área de trabajo donde está almacenada la cadena temporal, de forma que siempre se añade el primer carácter de la cadena.

Listado de «corriente 14-z$»

```
7995 REM ************************
7996 REM * RUTINA DE LA CORRIEN*
         * TE 14-Z$            *
7997 REM ************************
7998 REM st=POSICION INICIAL
7999 REM RECOMENDADA:65260    324
90
8000 RESTORE 8070
8005 LET c=0
8010 FOR i=st TO st+100
8020 READ a: POKE i,a: LET c=c+a
8030 NEXT i
8035 IF c<>10557 THEN  PRINT "ER
ROR EN SUMA DE CONTROL": STOP
8040 CLOSE #14
```

```
8050 RANDOMIZE USR st
8060 RETURN
8070 DATA 42,83,92,43,197,229,1,
11
8075 DATA 0,205,85,22,209,33,59,
0
8080 DATA 193,9,213,235,115,35,1
14,35
8085 DATA 235,1,247,255,9,1,9,0
8090 DATA 237,176,225,35,237,75,
79,92
8095 DATA 167,237,66,34,50,92,1,
0
8100 DATA 0,201,196,21,90,40,0,4
0
8105 DATA 0,11,0,245,42,75,92,12
6
8110 DATA 254,90,40,11,254,128,2
02,112
8115 DATA 6,205,184,25,235,24,24
0,35
8120 DATA 78,35,70,3,197,229,9,2
05
8125 DATA 82,22,35,235,225,193,1
12,43
8130 DATA 113,241,18,167,201
```

Capítulo 2

INICIACION AL EMPLEO DE LOS MICRODRIVES

Cómo funcionan

Cada cartucho de Microdrive contiene un bucle continuo de aproximadamente cinco metros de cinta magnética de calidad superior a la de las cintas de audio normales. En sí los Microdrives están formados por un motor que hace girar la cinta y una cabeza de grabación/reproducción parecida a la de un magnetófono de cassette. La cinta pasa por delante de la cabeza a gran velocidad y se pueden almacenar datos en ella mediante grabación o bien se pueden extraer al reproducir su contenido. Se trata de un sistema de cinta realmente rápido que está libre de muchas de las engorrosas operaciones típicas de los cassettes: rebobinado, etc.

El sistema también tiene sus inconvenientes. El principal es su tiempo de acceso. Si la parte de la cinta en que estaba guardado uno de nuestros programas acaba de rebasar la cabeza lectora y deseábamos leer precisamente ese programa, habrá que esperar a que toda la cinta vuelva a pasar para poder encontrarlo otra vez puesto que no es posible realizar un «rebobinado» de la cinta. Esto significa que puede necesitarse más de siete segundos para encontrar en la cinta un programa o un dato determinado.

La escritura de datos en un cartucho también puede resultar bastante lenta. Cuando el cartucho es nuevo y guarda pocos programas en su interior es fácil para el ordenador encontrar un lugar libre donde almacenar los nuevos datos o programas. En cambio, si el cartucho está bastante lleno, el ordenador tendrá que ir localizando los espacios libres y almacenando en ellos secciones del programa o de los datos que se deseaban almacenar. Esta circunstancia hará que la próxima vez que se desee leer esa información se requiera también más tiempo.

Protección contra escritura

Las cassettes de audio tienen en su parte posterior dos pequeñas pestañas o lengüetas que se pueden romper. Cuando esto se hace, el cassette queda protegido de forma que no se puede grabar nada en él. Así se preservan grabaciones que, por un motivo u otro, conviene conservar intactas. Estas lengüetas se denominan «lengüetas de protección contra escritura (o grabación) y borrado involuntarios». Los cartuchos para los Microdrives desarrollados por Sinclair también disponen de unas lengüetas de este tipo con una misión similar.

Tal y como se ve en la **figura 2** cada cartucho tiene una pequeña lengüeta de plástico en uno de sus lados. Si usted la levanta, impedirá la grabación de datos en ese cartucho. Esta es una posibilidad muy importante especialmente para los programas de tipo comercial (o para aquellos que a usted le interese conservar). Si, una vez separada la pestaña, usted desea grabar información en el cartucho, puede hacerlo cubriendo previamente el lugar ocupado por la lengüeta con un poco de cinta adhesiva. Hay que cuidar que la cinta adhesiva no sea demasiado ancha pues, si lo fuera y la dobláramos sobre la parte inferior y superior del cartucho, podría muy bien ocurrir que éste no entrara en el Microdrive.

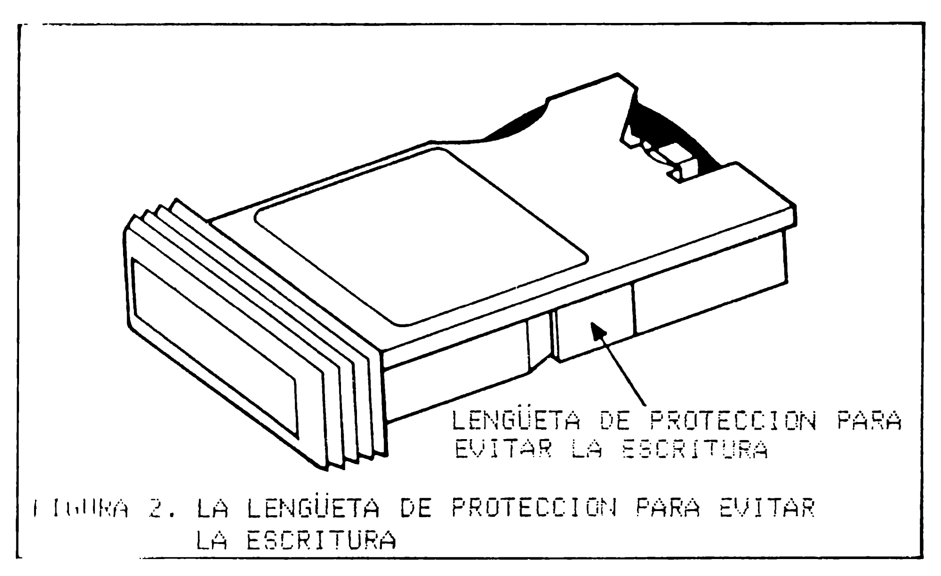


FIGURA 2. LA LENGÜETA DE PROTECCION PARA EVITAR LA ESCRITURA

Manejo de los cartuchos

La cinta contenida en el interior de los cartuchos es muy delicada. No hay que tocarla jamás ni con los dedos ni con otros objetos. Cuando no se utilice, es mejor dejar el cartucho dentro de la caja correspondiente, para protección.

Al insertar los cartuchos en los Microdrives asegúrese de que los coloca de la forma correcta (con la etiqueta grande en la parte superior) y empújelos con firmeza hacia adentro. Una vez los haya insertado asegúrese de que los ha introducido hasta el fondo pues, de otra forma, se podrían producir errores al comenzar a funcionar. Hay dos reglas muy importantes que deben seguirse al pie de la letra cuando se utilicen los cartuchos:

1. **Nunca hay que intentar sacar el cartucho si el LED rojo está encendido.**
2. **Nunca hay que dejar un cartucho en el Microdrive cuando se conecta o desconecta la alimentación.**

Si se deja de cumplir alguna de estas reglas se pueden causar daños permanentes, bien al cartucho o bien al propio Microdrive.

Los cartuchos no tienen la misma duración que los cassettes de audio normales debido a las duras condiciones bajo las que tienen que trabajar. Haga copias de seguridad de los programas importantes y guárdelos en otro cartucho o, mejor, en un cassette.

Inicialización de un cartucho mediante FORMAT

Cuando se adquiere un cartucho nuevo, virgen, la cinta de su interior contiene información magnética aleatoria. Si se intenta utilizar el cartucho se obtendrá un error del tipo **Microdrive not present** (Microdrive inexistente). Para poder utilizar el cartucho es necesario inicializarlo o «formatearlo» mediante el comando FORMAT. Esta instrucción tiene la forma general:

```
FORMAT "m";d;"NOMBRE"
```

donde «d» es el número del Microdrive (del 1 al 8) y «nombre» es el título que quedará permanentemente asociado con el cartucho.

Este nombre puede ser de hasta 10 letras de longitud. La inicialización del cartucho requerirá aproximadamente 30 segundos, tiempo durante el cual el ordenador formateará el cartucho, almacenará en él su nombre y comprobará cada fragmento de cinta varias veces. Si durante esta inspección el ordenador encuentra alguna parte de la cinta inutilizable porque, por ejemplo, ha resultado dañada, la marcará para evitar que se hagan operaciones sobre ella en el futuro. El borde de la pantalla se mostrará intermitente durante este proceso. Antes de pasar a formatearlo, asegúrese de que el cartucho está correctamente colocado en el Microdrive. La primera vez que formateé un cartucho me olvidé de empujarlo hasta el fondo: ¡el ordenador pensó que tan sólo la tercera parte del cartucho era utilizable! Cuando, después de insertarlo a fondo, volví a repetir la instrucción, el cartucho resultó formateado sin ningún problema. Con cartuchos recién adquiridos conviene hacer dos comandos FORMAT en rápida sucesión.

El comando FORMAT destruye cualquier programa que estuviera grabado en el cartucho. Así pues es mejor tomar precauciones antes de poner en marcha el formateo de un cartucho. Si se intenta formatear un cartucho al que se le ha quitado la lengüeta de protección contra escritura, se producirá un error.

Después de realizar FORMAT, se puede comprobar cómo ha funcionado el proceso entrando **CAT d** (donde **d** indica el número del Microdrive). Aproximadamente siete segundos después de haber entrado esta instrucción aparecerá el nombre del cartucho y luego un número. Este último indica la cantidad de kilobytes (K) utilizables que hay en el cartucho. Después de haber hecho FORMAT este número acostumbra a ser al menos 85. La mayor cantidad de kilobytes utilizables que he conseguido ha sido de 92.

Programas y Microdrives

Una de las ventajas principales de los Microdrives es su gran velocidad de almacenamiento y lectura de cassettes. Para utilizarlos se requiere emplear comandos similares a los empleados para los cassettes con unas pocas modificaciones—. Por ejemplo, para al-

macenar en cassette el programa de las corrientes que hemos visto en el capítulo anterior se podría hacer:

SAVE "corrientes"

Para emplear estos comandos con los Microdrives habría que insertar varias cosas en la línea inmediatamente posterior al comando. Primero, habría que añadir un asterisco (*) y luego una «**m**»; para indicar que se trata de una operación que afecta a un Microdrive. Luego habría que poner **d**; para especificar a qué Microdrive va a afectar la instrucción. Para guardar el programa "corrientes" en un cartucho colocado en el Microdrive 2 se podría hacer:

```
SAVE *"m";2;"CORRIENTES"
```

(se puede utilizar "m" o "M", no hay diferencia entre ellas). Los comandos que sigan a ésta no serán aceptados por el Spectrum a menos que el Interface esté conectado.

Cuando se pone en marcha una operación que afecta a cualquier Microdrive, el motor empieza a girar y el LED rojo se ilumina. Si el Microdrive está vacío, se producirá un sonido agudo y aparecerá el mensaje de error Microdrive no presente **(Microdrive not present)**. Normalmente, se puede saber si el conjunto Microdrive/cartucho funciona correctamente por el sonido que se produce. Cuando todo va bien se oye un suave rumor procedente del Microdrive. Si algo va mal —por ejemplo no se ha insertado el cartucho correctamente— se oye un golpeteo acompañado de un chirrido. Si esto ocurre, hay que empujar firmemente el cartucho hacia el interior del Microdrive. Si el ruido persiste, hay que hacer BREAK (pulsando CAPS SPACE), examinar el cartucho, colocarlo otra vez y probar de nuevo. Si el Microdrive chirría cuando tiene un cartucho colocado y aparece un mensaje de error, lo más probable es que el cartucho sea defectuoso y la cinta no pueda girar. Hay que sustituir el cartucho por otro. Si todo está bien, durante una operación SAVE el Microdrive funcionará durante aproximadamente siete segundos, mientras se está buscando el nombre del fichero especificado. Si se encuentra, aparecera el error «**Writing to a "read"**

file» (Escritura sobre un fichero de lectura). El borde de la pantalla parpadea siempre que se están escribiendo datos en un cartucho.

Al igual que con los cassettes, es posible emplear la función LINE dentro de una sentencia SAVE para que, automáticamente, un programa en BASIC pueda empezar inmediatamente después de haber sido cargado desde el cartucho. Para hacer que el programa anterior empiece en la línea 1000 se puede introducir:

```
SAVE *"m";2;"CORRIENTES"LINE 100
0
```

Después de realizar una instrucción SAVE se puede comprobar el correcto funcionamiento del sistema con la función «VERIFY»:

```
VERIFY *"m";2;"CORRIENTES"
```

Si aparece cualquier diferencia se producirá el mensaje **Verification failed** (La verificación ha fallado). Todos los programas referentes a los Microdrives no deben dar ningún error de comprobación. En caso contrario hay que comprobar a fondo tanto el cartucho como el Microdrive. Si el error persiste, hay que substituir uno o ambos dispositivos.

Para cargar un programa desde un cartucho se emplea el comando LOAD con el siguiente formato:

```
LOAD *"m";d;"NOMBRE"
```

donde «d» es el número del Microdrive y «nombre» es el nombre del programa que se solicita. Al contrario de lo que ocurre cuando se emplean cassettes, no es posible cargar el primer programa de la cinta utilizando como nombre del fichero una cadena vacía. Por ejemplo, si se intenta hacer

```
LOAD *"m";1;""
```

producirá el error **Invalid filename** (Nombre de fichero no válido). Si no se puede encontrar en el cartucho el nombre del programa,

se recibe el mensaje de error **File not found** (Fichero no encontrado).

La función de fusión («merge») también se puede emplear con los Microdrives y tiene un formato similar al de los anteriores comandos:

```
MERGE *"m";d;"NOMBRE"
```

Así se cargará el programa de nombre «nombre» desde el Microdrive «**d**» y lo refundirá con el programa y las variables ya existentes substituyendo las anteriores por las que están en el cartucho. Al contrario que con los cassettes, no se puede hacer MERGE con un programa que se ejecuta a sí mismo y que haya sido almacenado con LINE xxx. Si se intenta hacer esto, se obtiene el error **Merge error** (Error de fusión).

Un comando adicional que resulta necesario para emplear los cartuchos pero que no se requiere con los cassettes, es ERASE. Su misión es la de borrar del cartucho cualquier fichero. Su formato es:

```
ERASE "m";d;"NOMBRE"
```

Este comando no necesita un asterisco como es normal en otros referentes al Microdrive. No se puede colocar nada más después del nombre del fichero (por ejemplo, no se puede poner LINE 2000).

Si se hace BREAK mientras se está ejecutando una instrucción SAVE, el fichero quedará incompleto y no podrá ser cargado. Hay que borrar (mediante ERASE) todo fichero que haya quedado incompleto al haber interrumpido una instrucción SAVE. Si se intenta guardar mediante SAVE un programa que ya está en el cartucho, se obtiene el error **«Writing to a "read" file»** (Escritura sobre un fichero de lectura). Antes de cargar con SAVE un nuevo programa hay que borrar mediante ERASE el programa anterior.

Encadenamiento de programas

Utilizando SAVE *"m" ... LINE un programa puede empezar por sí mismo y cargar otro, por ejemplo, uno escrito en código máquina. El problema es que los programas encadenados sólo funcio-

narán en un Microdrive determinado. Para solucionar este problema, el primer programa debería tener una línea como la siguiente:

```
LET d=PEEK 23766
```

así la variable **d** contendrá el Microdrive desde el que se cargará el programa. Si llamáramos "prog2" al segundo programa, para cargarlo desde el primero tendríamos que hacer:

```
100 LOAD *"m";d;"PROG2"
```

Los dos programas funcionarán correctamente sin importar en qué Microdrive estén. Por cierto, las instrucciones de la forma:

```
100 LOAD *"m";PEEK 23766;"PROG2"
```

no funcionarán correctamente por razones técnicas. Empléese siempre una variable.

Almacenamiento de vectores y código máquina en cartucho

Además de almacenar programas en cartucho se pueden guardar vectores, código-máquina y el contenido de una o varias pantallas. Para ello, hay que colocar en los comandos correspondientes, determinadas palabras clave después del nombre del fichero. Para almacenar vectores, hay que colocar, después del nombre del fichero, la palabra DATA y luego el nombre del vector ya sea numérico o de elementos de tipo cadena. Por ejemplo, para guardar el vector b( ) en el Microdrive 2 bajo el nombre de "sueldos" habría que hacer:

```
SAVE *"m";2;"SUELDOS"DATA b()
```

Es posible guardar variables de tipo cadena normales no dimensionadas en cassette pero cuando se cargan de nuevo desde el cassette se degradan. En los Microdrives se ha evitado este problema. Para comprobarlo introduzca:

```
LET a$="COMPROBACION":SAVE *"m";
1;"CADENA" DATA a$()
```

que dará el error **Nonsense in BASIC** (Sin sentido en BASIC).

Las instrucciones en código-máquina y los bytes se pueden almacenar añadiendo la palabra clave CODE seguida de dos números separados por una coma. El primer número es la dirección inicial y el segundo el número de bytes. Ambos son opcionales en las sentencias LOAD y VERIFY. También es posible almacenar gráficos utilizando SCREEN$. Se pueden verificar mediante VERIFY los gráficos así guardados, cosa imposible con los cassettes.

Si se intenta almacenar un fichero utilizando un comando incorrecto se recibirá el error **Wrong file type** (Tipo de fichero inadecuado). Por ejemplo, ocurriría esto con:

```
LOAD *"m";1;"TEST"CODE
```

si «test» fuera un programa.

CATalogo

Para saber qué ficheros se tienen en un cartucho se debe emplear el comando CAT (abreviatura de «catálogo»). Este comando tiene la forma:

```
CAT d
```

donde **«d»** es el número del Microdrive. Por efecto de esta instrucción aparece en pantalla el nombre del cartucho, luego una línea en blanco, la lista alfabética de todos los ficheros que hay en ese cartucho y, finalmente, un número. Este número indica cuántos kilobytes (K) quedan libres en el cartucho. Si el cartucho está vacío la cifra será igual o mayor a 85. Se puede enviar el catálogo a otros canales además de a la pantalla mediante la instrucción:

```
CAT #n,d
```

donde **«n»** es el número de la corriente. Por ejemplo,

```
CAT #3,2
```

llevará el catálogo del Microdrive 2 a la corriente 3 que, normalmente es la impresora ZX. Con esta instrucción sólo se listan los primeros 50 ficheros que se encuentran. Si usted tiene más de 50 ficheros no podrá listarlos todos con la instrucción CAT.

¿Qué ocurre cuando, desde un programa, se desea conocer el número de ficheros que hay en un cartucho? Una forma de conseguirlo —aunque un tanto rudimentaria— consiste en borrar la pantalla, hacer CAT, y luego leer uno a uno cada carácter de la pantalla mediante la función SCREEN$. El método resulta engorroso y, en caso de que haya más de 20 ficheros en el cartucho, inútil (puesto que la pantalla se desplazaría hacia arriba y se perderían los nombres de varios ficheros). Lo mejor es emplear la rutina corriente **14-z$** para que vaya añadiendo caracteres a la variable **z$** y luego emplear:

```
LET z$="":CAT #14,d
```

De esta forma en la variable **z$** estará todo el catálogo que podrá así examinarse a voluntad. Para poder hacer uso de la información guardada en **z$** es necesario conocer cómo está organizada, es necesario conocer su formato. Es el siguiente:

Primero hay 10 caracteres que son el nombre del cartucho, seguidos de dos códigos correspondientes a NEWLINE (CHR$ 13s). Luego vienen todos los nombres de los ficheros ordenados alfabéticamente.

Después hay otro NEWLINE y luego uno o dos caracteres que indican el número de kilobytes libres. Finalmente hay otro NEWLINE.

El formato se puede apreciar mejor gracias a la siguiente subrutina que presenta en pantalla un catálogo en limpio. El código máquina de corriente **14-z$** debería introducirse e inicializarse antes de hacer GO SUB 2000.

Listado del catálogo en limpio

```
1999 REM CATALOGO EN LIMPIO
2000 LET z$="": CAT #14,d
2010 CLS
2020 PRINT "NOMBRE DEL CARTUCHO:
"; INVERSE 1;z$( TO 10)
2025 PRINT
2030 LET z$=z$(13 TO )
2040 LET f=0
2050 IF LEN z$<10 THEN  GO TO 21
00
2060 LET f=f+1
2070 PRINT f;". ";z$( TO 10)
2080 LET z$=z$(12 TO )
2090 GO TO 2050
2100 PRINT "QUEDAN ";''f;" FICHE
ROS";z$(2 TO LEN z$-1);"K"
2110 RETURN
```

Almacenamiento de variables

Si usted tiene un programa de cierto tamaño que utilice bastantes vectores y variables de las que no pueda prescindir, puede hacérsele muy lento almacenar todo el programa y sus variables (con un SAVE normal) o bien almacenar cada vector bajo un nombre de fichero distinto (usando SAVE ... DATA). Las siguientes subrutinas le permiten guardar en un cartucho todas las variables de un programa para luego poderlas descargar más tarde y, quizás, transferirlas a otro programa. Las variables se almacenan en el Microdrive 1 bajo los nombres «var1» y «var2», pero esto puede cambiarse fácilmente alterando las líneas 4040, 4050, 4110 y 4130. No es posible guardar nombres de ficheros y los números de los Microdrives en variables dada la técnica que se emplea para ello. Por cierto, que no hay ningún error: las líneas 4010 y 4015 son iguales pero ambas necesarias. La rutina de almacenamiento (SAVE) puede llamarse con GO SUB 4000 pero la rutina de carga (LOAD) tiene que ponerse en marcha con GO TO 4100 (no se puede emplear GO SUB puesto que contiene una sentencia CLEAR que anula to-

dos los GO SUBs previos). Es necesario adaptar la línea 4140 para que retroceda a la línea adecuada del programa que ha llamado a la rutina.

Listado de la rutina de almacenamiento y carga de variables

```
3999 REM ALMACENAMIENTO (SAVE)
        DE VARIABLES
4000 DEF FN p(p)=PEEK p+256*PEEK
 (p+1)
4010 LET l=FN p(23641)-FN p(2362
7)
4015 LET l=FN p(23641)-FN p(2362
7)
4020 POKE 23565,l-256*INT (l/256
)
4030 POKE 23566,INT (l/256)
4040 SAVE *"m";1;"var1"CODE 2356
5,2
4050 SAVE *"m";1;"var2"CODE FN p
(23627),l
4060 RETURN
4098
4099 REM CARGA (LOAD) DE
        VARIABLES
4100 CLEAR
4110 LOAD *"m";1;"var1"CODE 2356
5
4120 DIM a$(FN p(23565)-7)
4130 LOAD *"m";1;"var2"CODE FN p
(23627)
4140 GO TO xxxx: REM RESTO DEL
PROGRAMA
```

Copias múltiples

Tal y como ya hemos mencionado, es recomendable tener copias de seguridad en cartuchos o cassettes. Sin embargo, el Spectrum dispone de una instrucción —no recogida en el manual— que

permite almacenar un mismo fichero repetidas veces en el mismo cartucho con el mismo nombre. Por contra, si se intenta almacenar un fichero desde el BASIC dos veces consecutivas, bajo el mismo nombre, se provocará un error. Para conseguir múltiples copias de un programa, o de cualquier otro tipo de fichero, hay que hacer inmediatamente antes del SAVE un POKE 23791, x donde "x" es el número de copias (de 1 a 255). El valor de «x» que normalmente da los mejores resultados es el 2. Tras hacer esto, y después de haberse realizado el SAVE, el fichero se habrá copiado «x» veces. El número de copias se inicializa nuevamente a 1 después del SAVE. Usted no se dará cuenta de esto puesto que la instrucción CAT sólo imprimirá el nombre del fichero una sola vez. La ventaja de este método de obtención de copias múltiples es que si usted, o cualquier otra persona, borra accidentalmente el fichero sólo una de las copias resultará afectada. El resto de las copias permanecerá en el cartucho y podrán cargarse normalmente. Si se ha realizado «x» copias de un fichero para borrarlo se tendrá que entrar la instrucción ERASE «x» veces. Lógicamente las copias múltiples ocupan «x» veces más de espacio en el cartucho. Téngase muy en cuenta que las copias múltiples no están protegidas de los efectos de un comando FORMAT "m"; que, de una sola vez, las borrará todas.

Capítulo 3

GESTION DE FICHEROS DEL MICRODRIVE

Ficheros secuenciales

Un fichero es una estructura de información de la que se pueden leer o a la que se pueden añadir datos. Los ficheros secuenciales obligan a leer y escribir los datos según un orden determinado. Por ejemplo, si se deseara leer el décimo elemento de un fichero secuencial habría que leer primero los nueve que le preceden. En el Spectrum los ficheros se almacenan, normalmente, en un cartucho y son del tipo secuencial. (Los ficheros de **acceso directo** —el segundo tipo— no se pueden utilizar en BASIC).

En el Capítulo 1 se explicó cómo funcionan las corrientes y se describieron los canales «K», «S» y «P». Para almacenar datos en el Microdrive mediante cualquier procedimiento que no sea el de guardarlos en forma de programas o de vectores hay que designar al Microdrive mediante el identificador de canal —"m" (o "M") (por «Microdrive»). Análogamente, se debe recurrir a la sentencia OPEN # para asignar el canal «m» a una determinada corriente aunque, en este caso particular, la sintaxis de esa sentencia sea algo diferente por necesitar el Spectrum más información. La sentencia OPEN # tiene ahora la forma:

```
OPEN #n;"m";d;"NOMBRE"
```

la cual creará un nuevo canal con el especificador "m" en la unidad de cartuchos **d** bajo el nombre **"nombre"** y, luego, asignará ese nuevo canal a la corriente número **n**. Como el fichero **"nombre"** acaba de ser creado no puede contener aún dato alguno. Por esta razón se denomina fichero de **escritura.**

Ficheros de escritura

Este tipo de ficheros está formado por aquellos ficheros que no existían con anterioridad a la ejecución de la sentencia OPEN # y que han sido creados precisamente por ella. La sentencia principal para enviar datos a un fichero del Microdrive es la PRINT # n; que funciona de forma parecida a como lo hace la instrucción PRINT normal con la diferencia de que ahora los datos se «imprimen» en un cartucho. Como demostración, ejecútese el programa siguiente y obsérvese cuándo funciona realmente el Microdrive:

```
100 OPEN #4;"m";1;"FICHERO PRUEB
A"
110 FOR i=1 TO 500
120 PRINT i;"";
130 PRINT #4;i
140 NEXT i
150 CLOSE #4
```

Es posible que se haya pensado que el Microdrive está en funcionamiento la mayor parte del tiempo —cada vez que imprime un número—. ¡No es así! Después de la sentencia OPEN # la unidad de cartucho funciona durante el tiempo que necesita para buscar un fichero del mismo nombre que el especificado en la instrucción. No podrá encontrarlo ya que se trata de un fichero de escritura. Lo que hace OPEN # es crear un área de memoria llamada **«buffer»** (registro temporal) utilizada como zona de almacenamiento temporal o intermedio de caracteres. Cuando se encuentra una instrucción PRINT se almacenan los caracteres en el «buffer» y no se transferirán al cartucho hasta que el «buffer» no esté lleno. En ese momento, la unidad de cartucho funcionará durante aproximadamente un segundo (tiempo necesario para transferir los 512 caracteres del «buffer» al cartucho); luego, una vez vaciado el contenido del «buffer» lo dejará dispuesto para almacenar nuevos caracteres. Por esta razón, durante la ejecución del anterior programa, la unidad de cartucho sólo actuó en un par de ocasiones (el «buffer» se llenó unas cuatro veces). Es absolutamente imprescindible emplear una sentencia CLOSE # al trabajar con ficheros de escritura. Si no se hiciera

CLOSE, los últimos datos enviados al «buffer» no podrían llegar jamás al cartucho puesto que el «buffer» podría no haberse llenado totalmente. De esta forma, se obtendría un fichero incompleto. La sentencia CLOSE # hace que los datos que aún hay en el «buffer» pasen al cartucho y luego elimina el «buffer» de la memoria. Si se ejecuta una sentencia CLEAR # mientras está abierta una corriente de escritura, el fichero resultará incompleto y no será posible utilizarlo. Un fichero parcial de este tipo debe ser borrado. Si se intenta abrir (con OPEN) un fichero que ya está abierto para otra corriente se obtiene el mensaje de error **Reading from a "write" file** (Lectura de un fichero de escritura).

Ficheros de lectura

Cuando ya se ha abierto con OPEN #, se le han enviado datos y ha sido cerrado mediante CLOSE el fichero de escritura pasa a ser de "lectura" y aparece en el catálogo del cartucho. Para leer los datos contenidos en ese fichero hay que abrirlo mediante la sentencia OPEN # con la misma sintaxis que antes. La lectura de los datos en sí se realiza normalmente con la sentencia INPUT #. Para leer los datos existentes en el fichero creado por el programa anterior introdúzcase:

```
100 OPEN #4;"m";1;"FICHERO PRUEB
A"
110 INPUT #4;i
120 PRINT i;" ";
130 GO TO 110
```

Estas sentencias crean inicialmente un canal de ficheros de lectura en la corriente 4. Luego, se lee del fichero la variable **i** (línea 110) que se imprime en la pantalla. El programa, retrocede entonces a su inicio y vuelve a empezar. Terminará con un error de **End of File** (Fin de fichero). Veremos más adelante en este mismo Capítulo cómo detectar el fin de un fichero antes de que se produzca. Como probablemente se habrá adivinado, mientras se ejecutaba el programa

también se emplea un «buffer» cuando se leen datos desde un fichero. Este «buffer» tiene igualmente una capacidad de 512 caracteres. Cuando se encuentra una sentencia INPUT # se lee un bloque de 512 caracteres (o menos si se trata del último bloque) desde el cartucho y se almacena en el «buffer». Cuando una rutina de entrada (o una sentencia INKEY$ #) requiere un carácter lo extrae del «buffer». Si es el último que allí había, se leerá otro bloque de 512 caracteres del cartucho. El proceso se repetirá hasta que no haya nada más en el fichero produciéndose entonces un error de **Fin de fichero.**

Para leer caracteres de un fichero, además de utilizar la sentencia INPUT #, se puede recurrir a la INKEY$ # que ya mencionamos en el capítulo 2. La instrucción INPUT va reuniendo los caracteres en secuencia hasta que alcanza el final de una línea y, en ese momento, asigna la secuencia a una variable. Por el contrario, INKEY$ #n sólo lee un carácter. A diferencia de la función INKEY$ normal, INKEY$ # «espera» al carácter. (Los INKEY$ # aplicados a una corriente del Microdrive no dan nunca como resultado una cadena vacía [" "]). Una vez se ha acabado de leer un fichero hay que hacer una sentencia CLOSE #. Aunque esto no es tan importante como con los ficheros de escritura, resulta aconsejable por razones de elegancia y para evitar que haya grandes cantidades de memoria ocupadas por «buffers» vacíos. Se puede hacer perfectamente CLEAR # con ficheros de lectura abiertos.

Cuando se utiliza la sentencia INPUT # el contador empleado por el Spectrum para hacer aparecer la petición «scroll?» en la pantalla se actualiza de forma que, si se lee el contenido de un fichero y se imprime en la pantalla, ésta se desplazará en sentido vertical continuamente. Si no interesa obtener este efecto, se puede insertar en el programa una línea como ésta:

```
LET sc=PEEK 23692: INPUT #(LO Q
UF SEA): POKE 23692,sc
```

Cualquier sentencia INPUT # actualiza el contador aunque normalmente no es posible apreciarlo. Una forma sencilla de eliminar la petición de «scroll» en un programa es utilizar la sentencia:

```
INPUT ""
```

Algunos comentarios acerca de PRINT e INPUT

Hay que ser muy precavido al emplear las sentencias PRINT # e INPUT # con los ficheros del Microdrive por los efectos que pueden tener los separadores situados entre los argumentos. Cuando se imprima sobre la pantalla o con la impresora y se emplee una coma (,) se saltará al principio de la siguiente media línea. Si se coloca un apóstrofe ('), a su vez, se pasa a una nueva línea. Ahora bien, en los ficheros del Microdrive no existen «líneas» y, si se coloca una coma en una sentencia que imprima datos en un fichero (por ejemplo, PRINT # 4; a, b), el Spectrum enviará, en vez del valor de las variables, el código de control que corresponde a la coma (CHR$ 6). De forma similar, el apóstrofe hará que se envíe un CHR$ 13 (correspondiente a NEWLINE). Los CHR$ 6 pueden confundir al Spectrum cuando se emplee INPUT # (aunque no ocurrirá esto si se recurre a INKEY$). Como ejemplo, pruébese el siguiente programa:

```
  10 OPEN #4;"m";1;"PRUEBA ERROR"
  20 LET a$="PRIMERO": LET b$="S
EGUNDO"
  30 PRINT #4;a$,b$
  40 CLOSE #4
  50 OPEN #4;"m";1;"PRUEBA ERROR"
  60 INPUT #4;a$
  70 INPUT #4;b$
  80 CLOSE #4
```

Quizás no se esperaba el error de **Fin de fichero** de la línea 70. Sin embargo, el PRINT # de la línea 30 envía el texto «primero» seguido de la coma (CHR$ 6), el texto «segundo» y, finalmente, un NEWLINE (CHR$ 13). La sentencia INPUT # supone que las variables guardadas en un fichero están separadas mediante CHR$ 13. Consecuentemente, en este programa INPUT # asignó a a$ «primero» + CHR$ 6 + «segundo» y luego vació el fichero. Al in-

tentar asignar a b$ un carácter contenido en el fichero, se produjo el error de **Fin de fichero.**

Los separadores de las sentencias INPUT # también crean otros problemas poco previsibles. Si se hace:

```
10 OPEN $4;"m";1;"FICHERO PRUEBA
S"
20 INPUT #4;a$,b$
```

(suponiendo que «fichero prueba» exista) se recibirá el error **Writing to a «read» file** (Escritura sobre un fichero de lectura). La explicación está en que la coma de la línea 20 intenta enviar un CHR$ 6 a un fichero de lectura. Si en un fichero hay uno o varios signos de comillas (") pueden aparecer problemas cuando se intenten leer cadenas mediante una sentencia INPUT #. Por tanto, conviene emplear sentencias del tipo INPUT # ... LINE. Las reglas de oro para utilizar PRINT # e INPUT # son:

1. Con PRINT # se deben separar las variables mediante apóstrofes o bien emplear líneas separadas. Por ejemplo, **PRINT # 4; a$'b$** o **PRINT # 4; a$: PRINT # 4; b$.**
2. En las sentencias INPUT hay que emplear como separadores los puntos y comas. Cuando se usen variables de tipo cadena se debe recurrir a **LINE** por ejemplo **INPUT # 4; LINE a$; LINE b$.**

Empleo de MOVE con los ficheros del Microdrive

El comando MOVE tiene multitud de usos pero aquí trataremos únicamente de aquellos que están directamente relacionados con los ficheros del Microdrive. Si se dispone de un fichero de datos y se quiere ver qué contiene sin tener que abrirlo y leer carácter a carácter, se puede hacer:

```
MOVE "m";d;"NOMBRE"TO #2
```

(donde **TO** es una palabra clave) con lo que se imprimirá en la corriente 2 (la pantalla) el contenido del fichero que se especifique. Si se intenta hacer MOVE con otro tipo de fichero (por ejemplo, un pro-

grama) se obtendrá un error **Wrong file type** (Tipo de fichero erróneo). Si se desea obtener una copia sobre papel de un fichero se puede hacer:

```
MOVE "m";d;"NOMBRE"TO #3
```

que provocará la impresión sobre la corriente 3 (la impresora ZX). También se puede emplear MOVE para copiar ficheros de datos:

```
MOVE "m";d1;"FICHERO 1" TO "m";d
2;"FICHERO 2"
```

que transfiere los datos de «Fichero 1» de la Unidad de Cartucho 1 al «Fichero 2» de la Unidad de Cartucho 2.

Inserción de elementos en un fichero

Supongamos que disponemos de un fichero que contiene una gran cantidad de datos, como, por ejemplo, una lista de nuestros programas. Supongamos también que deseamos añadir nuevos elementos al final de ese fichero. Como no se puede escribir sobre un fichero de lectura, es necesario crear un nuevo fichero, transferir a él el contenido del antiguo fichero y, mientras el nuevo fichero está aún abierto, añadir los nuevos datos. Una forma de conseguir esto sería leyendo los elementos del anterior fichero uno a uno y escribiéndolos en el nuevo. Sin embargo, se trata de un método lento e ineficaz que, con algunos tipos de datos, puede no funcionar. La mejor solución consiste en utilizar el comando MOVE tal y como se demuestra en el siguiente programa que añade al fichero «software» los elementos "25 Diciembre 1983" e "Invasores del espacio":

```
10 OPEN #4;"m";1;"SOFTWARE2"
20 MOVE "m";1;"SOFTWARE" TO #4
30 PRINT #4;"INVASORES DEL ESPAC
IO","25 DICIEMBRE 1983"
40 CLOSE #4
```

Si se desea que el nuevo fichero tenga el mismo nombre que el antiguo se pueden añadir las líneas siguientes:

```
50 ERASE "m";1;"SOFTWARE"
60 MOVE "m";1;"SOFTWARE2" TO "m"
;1;"SOFTWARE"
70 ERASE "m";1;"SOFTWARE2"
```

El programa anterior se ejecutará mucho más rápidamente si los dos ficheros están almacenados en Microdrives distintos.

La instrucción MOVE resulta también muy útil para unir el contenido de dos ficheros de datos —por ejemplo para conseguir el fichero «tercero» a partir de los ficheros «primero» y «segundo»—:

```
10 OPEN #4;"m";1;"TERCERO"
20 MOVE "m";1;"PRIMERO" TO #4
30 MOVE "m";1;"SEGUNDO" TO #4
40 CLOSE #4
```

Al igual que antes, la velocidad de ejecución del programa puede aumentarse notablemente empleando dos o incluso tres Microdrives con un fichero en cada uno.

Empleo de LIST

Aunque la forma más normal de escribir datos en un cartucho es mediante la instrucción PRINT #, también puede utilizarse la sentencia LIST #. Esta instrucción listará un programa en BASIC sobre una corriente que puede ser un canal del Microdrive. Cuando se lee el fichero, la palabra especial (por ejemplo STOP) será de tan sólo un carácter (CHR$ 266 por STO) en vez de varios caracteres con espacios en blanco entre las palabras clave. Puede resultar muy útil convertir un programa a caracteres mediante LIST # si se desea recorrer el programa en BASIC con, por ejemplo, un procesador de textos. El programa siguiente —«Búsqueda»— buscará una determinada secuencia de caracteres en el fichero creado por las sentencias OPEN #, LIST # y CLOSE # e imprimirá en la pantalla toda línea que la contenga. Esta posibilidad resulta especialmente útil cuando se busca el nombre de alguna variable o una determinada sentencia GO TO. Se puede emplear también para buscar en cualquier tipo de fichero de datos que no sólo sean listados de programas. El programa termina con un mensaje de **End of file** (Fin de fichero).

```
999 REM PROGRAMA BUSQUEDA
1000 INPUT "UNIDAD CARTUCHO N.?"
;d;"NOMBRE FICHERO?" ; LINE f$
1005 CLOSE #4: OPEN #4;"m";d;f$
1010 INPUT "CADENA QUE BUSCAR?" ;
 LINE b$
1020 INPUT #4; LINE a$
1030 FOR i=1 TO LEN a$-LEN b$
1035 IF a$(i TO i+LEN b$-1)=b$ T
HEN PRINT a$: PAUSE 50: GO TO 10
50
1040 NEXT i
1050 GO TO 1020
```

Si se desea buscar palabras especiales (como GO TO) hay que pulsar THEN con lo que se pone al Spectrum en el modo «K» y, luego, pulsar la tecla correspondiente (G para GO TO) para, finalmente, borrar la palabra THEN.

EOF: End of file (Fin de fichero)

Al leer un fichero se acaba por alcanzar su final. Si se intenta leer más allá de ese punto se provoca un error de fin de fichero. Sin embargo, existen varios métodos para detectar el fin de un fichero antes de que ocurra, métodos que hacen los programas más elegantes y eficaces.

La forma más sencilla de detectar un EOF consiste en escribir y antes de hacer CLOSE un carácter o secuencia de caracteres que normalmente no debería utilizarse al final del fichero. Mis preferencias personales me hacen emplear la cadena CHR$ 0 + CHR$ 0 que nunca utilizo para otra cosa. Este método va muy bien para la mayoría de los programas pero no puede emplearse si se tiene un programa diseñado para leer todo tipo de ficheros de datos o que pueda leer ficheros de programas o de código máquina (cómo se consigue esto último se explicará más adelante).

Otros BASICs basados en disco tienen la función ON EOF GOTO o, más generalmente, ON ERROR GOTO que produce el salto a una determinada línea cuando se detecta un EOF (o cualquier otro error). Desgraciadamente, el Spectrum carece de esta función pero, con un par de líneas en BASIC, se puede detectar el 99 % de los EOFs.

Si se añaden las siguientes líneas a un programa (colocándolas lo más cerca posible del principio) se dispondrá de la función FN E(x) donde «x» es el número de una corriente. Esta función da por resultado el valor 1 si el fichero está vacío o el valor 0 en cualquier otro caso.

```
  10 DEF FN E(A)=((INT (PEEK (FN
 D(A)+67)/2)-2*INT (PEEK (FN D(A
)+67)/4) AND (FN P(FN D(A)+11))=
FN P(FN D(A)+69))))
  11 DEF FN P(P)=PEEK P+256*PEEK
 (P+1)
  12 DEF FN D(D)=FN P(D*2+23574)
+FN P(23631)-1
```

Si la corriente **x** no está abierta o no es un canal «M», entonces el resultado no tendrá sentido alguno. El 1 % de las veces en que esta función no trabaja correctamente corresponde a aquellos casos en los que el número de caracteres existentes en el «buffer» es un múltiplo exacto del número 512 lo cual, afortunadamente, pasa muy pocas veces. Para asegurarse de que no ocurre esto, hay que utilizar la función:

```
FN P(FN D(X)+11
```

mientras se crea un fichero de escritura y antes de cerrar la corriente «x». Si es 0 hay que imprimir con anterioridad un carácter extra en la corriente.

Se puede recurrir también a la siguiente rutina en código máquina para disponer de una función ON EOF GOTO. Su longitud es de 67 bytes. Puede situarse en cualquier lugar de la memoria. Antes de hacer GO SUB 8150 la variable **eof** ha de tener por valor el de la posición requerida, y **línea** el del número de línea requerido (que debe existir). La rutina carga el código mediante POKEs y luego lo ejecuta. Después de esto, en cuanto se da un EOF, no se produce ningún mensaje de error sino que el intérprete salta a la línea especificada. La función no actúa cuando se produce cualquier error (incluido EOF). Si usted quiere cambiar el número de línea del salto

provocado por el error, cambie **línea** y GOSUB 8220. Para saber en qué línea se ha producido un EOF.

```
LET LINERROR=PEEK 23753+256*PEEK
 23754
```

Hay que hacerlo antes de que se realice ninguna operación con el Microdrive pues, de lo contrario, se obtendrá un resultado incorrecto.

Listado de ON EOF GOTO

```
8145 REM ************************
8146 REM *   GOTO EOF (DE FIN   *
         *   DE FICHERO)        *
8147 REM ************************
8148 REM EOF=POSICION INICIO, LI
NEA=SALTO A ERROR
8149 REM RECOMENDADA:65190 / 324
90
8150 RESTORE 8250
8160 LET c=0
8170 FOR i=eof TO eof+66
8180 READ a: LET c=c+a
8190 IF a<>260 THEN  POKE i,a
8200 NEXT i
8210 IF c<>6456 THEN  PRINT "ERR
OR EN SUMA DE CONTROL": STOP
8220 POKE eof+49,linea-256*INT (
line/256)
8225 POKE eof+50,INT (linea/256)
8230 RANDOMIZE USR eof
8240 RETURN
8250 DATA 33,17,0,9,235,42,61,92
8260 DATA 115,35,114,207,49,1,0,
0
8270 DATA 201,42,61,92,58,58,92,
254
8280 DATA 7,194,3,19,94,35,86,21
3
```

```
8290 DATA 205,176,22,253,203,55,
174,205
8300 DATA 110,13,42,69,92,34,201
,92
8310 DATA 17,260,260,33,66,92,11
5,35
8320 DATA 114,35,54,1,253,54,0,2
55
8330 DATA 195,125,27
```

Empleo de programas etc. como ficheros de datos

Normalmente si se intenta abrir una canal del Microdrive con el nombre de un programa o de cualquier otro tipo de fichero que no sea de datos, se obtiene un error **Wrong file type** (Tipo de fichero incorrecto). Puede resultar muy conveniente leer ese tipo de ficheros desde el BASIC. Se puede conseguir esto empleando la siguiente rutina en código máquina a la que hemos dado el nombre de «open # cualquier cosa». Es prácticamente igual a la sentencia OPEN pero permite abrir cualquier tipo de fichero. También informa acerca de si el fichero existe o no lo que también resulta muy interesante. Antes del GO SUB 8350 la variable **abierto** debe tener el valor de la posición de memoria que debe ocupar la rutina (se recomienda la posición 32320 para 16 K y la 65090 para 48 K). Con el GOSUB se carga el código (mediante POKE) pero no se ejecuta.

Listado de OPEN # cualquier cosa

```
8344 REM ************************
8345 REM * OPEN#CUALQUIER COSA .*
8346 REM ************************
8347 REM open=POSICION DE INICIO
8348 REM RECOMENDADA:65090 / 323
20
8350 RESTORE 8400: LET c=0
8360 FOR i=open TO open+93
8370 READ a: LET c=c+a
```

```
8375 POKE i,a
8380 NEXT i
8385 IF c<>10725 THEN  PRINT "ER
ROR EN SUMA DE CONTROL": STOP
8390 RETURN
8400 DATA 58,216,92,205,39,23,33
,17
8410 DATA 0,175,237,66,1,0,0,216
8420 DATA 50,215,92,33,10,0,34,2
18
8430 DATA 92,42,123,92,34,220,92
,58
8440 DATA 216,92,135,33,22,92,95
,22
8450 DATA 0,25,217,229,217,229,2
07,34
8460 DATA 221,203,24,70,40,13,17
5,207
8470 DATA 33,207,44,225,217,225,
217,1
8480 DATA 1,0,201,221,203,4,190,
175
8490 DATA 229,207,33,209,225,115
,35,114
8500 DATA 221,126,67,230,4,198,2
,79
8510 DATA 6,0,217,225,217,201
```

Para utilizar esta rutina hay que indicarle qué unidad de cartucho, corriente y fichero nos interesa. Esta información se pasa haciendo POKE 23766 (unidad de cartucho), POKE 23768 (corriente) y POKE sobre las 10 primeras posiciones del área de gráficos definidos por el usuario para guardar el nombre del fichero. Por ejemplo, si se desea hacer OPEN # 6; "m"; 2; "test" (donde "test" es cualquier tipo de fichero) las instrucciones tendrán que ser:

```
3500 POKE 23766,2: REM NUMERO UN
IDAD
3510 POKE 23768,6: REM NUMERO CO
RRIENTE
```

```
3520 LET A$="TEST"
3530 FOR i=1 TO 10
3540 IF i>LEN a$ THEN  POKE USR
"a"+i-1,32: GO TO 3560
3550 POKE USR "a"+i-1,CODE a$(i)
3560 NEXT i
3570 LET a=USR open: REM LLAMAR
A LA RUTINA
```

Si el nombre del fichero es de menos de 10 caracteres hay que añadir el número necesario de espacios (CHR$ 32s) en blanco y luego transferirlos mediante POKE (línea 3540). El valor que retorna la función **USR open** (**a** en el ejemplo) significa lo siguiente:

0 — corriente ya abierta
1 — fichero no encontrado
2 — fichero de datos
6 — no se trata de un fichero de datos.

Si el fichero no existe, se cierra la corriente y no se crea el fichero. Por tanto, el método es únicamente válido para ficheros de lectura. La rutina no crea ficheros de escritura (al contrario de lo que ocurre con el comando OPEN #).

Si el número devuelto por la rutina es un 6 —es decir, si no se trata de un fichero de datos— ¿cómo se sabe de qué tipo de fichero se trata? La respuesta está en los nueve primeros caracteres leídos del fichero. Estos caracteres guardan todas las características del fichero entre las que se incluye el tipo. El primer carácter determina el tipo:

0 Programa en BASIC
1 Vector numérico
2 Vector de cadenas
3 Bytes

Los ocho bytes siguientes guardan detalles como la longitud, inicio, número de línea, etc. y, en realidad, se sacan de las variables del sistema HD_00 a HD_11 (Ver el Apéndice A para más detalles). Lo que viene tras los nueve primeros bytes es el fichero en

sí. El siguiente programa, «Catálogo completo», es una versión notablemente mejorada de la función CAT. Además de dar todos los nombres de los ficheros, da también todos los otros detalles referentes al fichero: su tipo, longitud, número de la línea de auto inicio, etc. El programa necesita las rutinas «Corriente 14-z$» y «Open cualquier cosa» para funcionar. Es algo más lento que CAT pero ofrece una gran cantidad de información útil.

Listado de «Catálogo completo»

```
4997 REM ************************
4998 REM *  CATALOGO COMPLETO   *
4999 REM ************************
5000 LET z$="": CAT #14,d
5010 CLS : POKE 23766,d: POKE 23
768,15
5020 PRINT  INVERSE 1;"NOMBRE:";
z$( TO 10)
5030 LET z$=z$(13 TO )
5040 IF LEN z$<10 THEN  GO TO 53
30
5050 FOR i=1 TO 10
5060 POKE USR "a"+i-1,CODE z$(i)
5070 NEXT i
5080 CLOSE #15: LET z=USR open
5090 PRINT z$( TO 10);"  ";
5100 IF z=2 THEN  PRINT "FICHERO
 DE DATOS": GO TO 5300
5110 LET a$="": FOR i=1 TO 9
5120 LET a$=a$+INKEY$#15: NEXT i
5130 LET z=CODE a$(1)
5140 GO TO 5150+z*40
5149 REM z=0 PROGRAMA
5150 PRINT "LINEA DE PROGRAMA ";
CODE a$(8)+256*CODE a$(9)
5160 GO TO 5300
5189 REM z=1  CONJUNTO NUMERICO
5190 PRINT "CONJUNTO  ";CHR$ (CO
DE a$(6)-32);"()"
5200 GO TO 5300
```

```
5229 REM z=2 CONJUNTO DE CADENAS
5230 PRINT "CONJUNTO  ";CHR$ (CO
DE a$(6)-96);"$()"
5240 GO TO 5300
5269 REM z=3  CODIGO
5270 PRINT "CODIGO  ";CODE a$(4)
+256*CODE a$(5);
5280 PRINT ",";CODE a$(2)+256*CO
DE a$(3)
5300 CLOSE #15
5310 LET z$=z$(12 TO )
5320 GO TO 5040
5330 PRINT 'z$(2 TO LEN z$-1);"K
 BYTES LIBRES"
5340 RETURN
```

Generación de ficheros que no guardan datos

Desde un programa en BASIC, además de leer desde el cartucho ficheros que no contengan datos, es posible generarlos. No hace falta recurrir al código máquina; basta un comando POKE. El empleo de esta posibilidad puede ser bastante difícil y no la recomiendo a los principiantes: los errores que se cometen en su utilización pueden ser fatales para el ordenador.

El método empleado consiste en colocar tras una sentencia OPEN # unas líneas como las siguientes:

```
 100 DEF FN p(p)=PEEK p+256*PEEK
 (p+1)
 110 POKE FN p(s*2+23566+8)+FN p
(23631)+66,4
```

donde **s** es el número de la corriente de la instrucción OPEN. Así se engaña al sistema haciéndole crear segmentos de ficheros que no son de datos cada vez que se escribe en el cartucho un dato que esté en la corriente. Después del POKE se deben escribir nueve caracteres en la corriente elegida. El primero indica el tipo del fichero que deseamos generar y los ocho siguientes determinan los otros parámetros del fichero y son los mismos que se han menciona-

do anteriormente para los ficheros de lectura: las variables del sistema HD__00 a HD__11 (Véase Apéndice A).

Como ejemplo, el programa siguiente almacena todas las variables de un programa en BASIC en un programa como el del Capítulo 2, pero con una importante diferencia: en vez de guardarlas como dos ficheros de tipo CODE las almacena como un fichero de programas de forma que puedan ser fundidas (MERGE) con otros programas. Este puede resultar un método más lento que el anterior pero es mucho más flexible.

Listado de «Almacenamiento de variables»

```
3999 REM ALMACENAMIENTO DE VARIA
BLES COMO PROGRAMA
4000 OPEN #4;"m";1;"variables"
4010 DEF FN p(p)=PEEK p+256*PEEK
 (p+1)
4020 LET d=FN p(4*2+23566+8)+FN
p(23631)-1
4030 IF PEEK (d+4)<>CODE "M" THE
N  PRINT "Error": STOP
4040 POKE d+67,4: REM arreglar
4045 FOR i=1 TO 2: NEXT i
4050 LET z=FN p(23641)-FN p(2362
7)-1
4055 LET z=FN p(23641)-FN p(2362
7)-1
4060 PRINT #4;CHR$ 0;: REM senal
izador del programa
4070 PRINT #4;CHR$ (z-256*INT (z
/256));CHR$ INT (z/256);
4080 PRINT #4;CHR$ PEEK 23627;CH
R$ PEEK 23628;: REM inicio
4090 PRINT #4;CHR$ 0;CHR$ 0;: RE
M longitud programa
4100 PRINT #4;CHR$ 255;CHR$ 255;
: REM 'numero de linea'
4110 FOR i=FN p(23627) TO FN p(2
3627)+z-1
4120 PRINT #4;CHR$ PEEK i;
```

```
4130 NEXT i
4150 CLOSE #4
4160 RETURN
```

La línea 4000 crea el fichero de escritura. La línea 4040 realiza el POKE correspondiente. Aunque las líneas 4045 y 4050 puedan parecer supérfluas son, en realidad, imprescindibles. La variable **z** guarda el número de bytes que hay en el área de variables (línea 4055). La línea 4060 imprime el primer carácter para indicar que el fichero es de programas. La línea 4070 imprime los dos caracteres que definen la longitud del fichero y la línea 4080 hace lo mismo con los dos que indican el inicio del área de variables. La línea 4090 asigna el valor 0 a la longitud del programa. La línea 4100 hace que la línea de auto inicio sea 65535 (es decir que no haya auto inicio). El bucle de las líneas 4110 a 4130 envía cada byte de la variable al fichero antes de que la línea 4150 lo cierre.

Otras formas de leer datos

Aunque, como ya hemos dicho, las sentencias INPUT # e INKEY$ # son las más usadas para leer ficheros, la MOVE puede resultar más útil. Si la rutina «Corriente 14-z$» está activa, la línea:

```
LET z$="" :MOVE "m" ;1 ; "FICHERO" T
O #14
```

leerá todo el fichero de datos de una sola vez desde el cartucho y lo guardará en la variable de tipo cadeza z$ (si la memoria lo permite). Una vez hecho esto, la variable puede ser manipulada y modificada según se requiera y luego puede escribirse en otro fichero con una sola instrucción PRINT, por ejemplo:

```
OPEN #4; "m" ;1 ; "FICHERO2" :PRINT #
4;Z$;: CLOSE #4
```

Este método es el que tiene una menor repercusión sobre el cartucho en cuanto a desgaste y desperfectos lo cual puede ser un

factor importante cuando se trabaje con ficheros de datos de gran volumen y uso frecuente (puesto que la unidad de cartucho sólo actúa dos veces: una para leer y otra para escribir).

Rutina de estado

La rutina siguiente, denominada «Estado», permite que un programa pueda saber si un Microdrive en particular está conectado o no y si hay un cartucho en él. En este último caso se indica si el cartucho está protegido contra escritura o no. Antes de hacer GOSUB 8550 la variable **stat** debe tener por valor el de la posición de memoria deseada.

Listado de «Estado»

```
8545 REM *******************
8546 REM *     ESTADO      *
8547 REM *******************
8548 REM stat=DIRECCION INICIO
8549 REM recomendada:64960 / 321
90
8550 RESTORE 8630: LET c=0
8560 LET x2=INT ((stat+100)/256)
: LET x1=stat-256*x2+100
8570 FOR i=stat TO stat+128
8580 READ a: LET c=c+a
8590 POKE i,a
8600 NEXT i
8610 IF c<>14341+4*(x1+x2) THEN
 PRINT "error en suma de control
": STOP
8620 RETURN
8630 DATA 58,214,92,243,24,33,33
,136
8640 DATA 19,43,125,180,32,251,3
3,136
8650 DATA 19,6,6,219,239,230,4,3
2
```

```
8660 DATA 4,16,248,24,84,43,124,
181
8670 DATA 32,239,1,0,0,24,84,17
8680 DATA 0,1,237,68,198,9,79,6
8690 DATA 8,13,32,20,122,50,247,
0
8700 DATA 62,238,211,239,205,x1,
x2,62
8710 DATA 236,211,239,205,x1,x2,
24,17
8720 DATA 62,239,211,239,123,211
,247,205
8730 DATA x1,x2,62,237,211,239,2
05,x1
8740 DATA x2,16,214,122,211,247,
62,238
8750 DATA 211,239,24,162,197,245
,1,135
8760 DATA 0,11,120,177,32,251,24
1,193
8770 DATA 201,219,239,230,1,1,1,
0
8780 DATA 32,1,3,197,175,207,33,
193
8790 DATA 201
```

La instrucción GO SUB sólo pone el código máquina en el lugar adecuado pero no hace que se ejecute. Antes de usarla hay que colocar el número de la unidad de cartucho mediante POKE 23766 y luego utilizar unas cuantas líneas como las siguientes:

```
2500 LET a=USR stat
2510 IF a=0 THEN  PRINT "MICRODR
IVE NO CONECTADO"
2520 IF a=1 THEN  PRINT "CARTUCH
O PRESENTE"
2530 IF a=2 THEN  PRINT "CARTUCH
O PROTEGIDO CONTRA ESCRI TURA"
```

Comandos para color

En el manual del Interface de Sinclair se hace de pasada referencia a la posibilidad de que los comandos para color puedan no funcionar después de emplear canales que no sean el «K», el «S» o el «P». De hecho, éste es un serio problema que hay que tener muy en cuenta cuando se manejan ficheros.

El problema consiste en que, después de utilizar los canales del Interface para salida, los comandos permanentes para color (como PAPER 3) no tienen efecto aparente. Y, lo que es más importante, lo que hacen tales comandos es enviar datos a ficheros de escritura. Para corregir este problema, antes de fijar los colores permanentes hay que hacer una sentencia **PRINT;**. Para ver este error en acción, haga lo siguiente:

```
10 OPEN #4;"m";1;"COLPRUEBA"
20 PRINT #4;"PRIMERA LINEA"
30 FLASH 1: PAPER 4: CLS
40 PRINT #4;"SEGUNDA LINEA"
50 CLOSE #4
60 MOVE "m";1;"COLPRUEBA" TO #2
```

La línea 10 crea el fichero de escritura «colprueba» y la 20 le envía una línea de datos. La línea 30 debería hacer que toda la pantalla se mostrara intermitentemente verde pero, en cambio, envía los códigos de color al cartucho. Las líneas 40 y 50 envían otra línea de datos al fichero y luego lo cierran. Finalmente, la línea 60 imprime todo el fichero en la pantalla con lo que es posible descubrir la indeseada inserción de los códigos de color. La forma de corregir el problema en este caso es hacer:

```
25 PRINT ;
```

Ampliación del «catálogo de corrientes»

Se pueden añadir las siguientes líneas al programa «Catálogo de corrientes» para que dé más detalles acerca de las corrientes del Microdrive.

```
1630 IF F$="M" THEN  GO TO 2500
1840 IF FN P(D)<>8 AND FN P(D+2)
<>8 THEN  RETURN
1850 PRINT "SALIDA DE LA ROM ADI
CIONAL:";FN P(D+5)
1860 PRINT "ENTRADA ROM
      :";FN P(D+7)
2499 REM CANAL M
2500 PRINT  INVERSE 1;"MICRODRIV
E"
2510 GO SUB 1800
2520 PRINT "UNIDAD NUMERO
      :";PEEK (D+25)
2530 LET M=FN P(D+26)
2540 PRINT "MAPA DE LA CINTA:"
2550 FOR I=0 TO 31: FOR J=1 TO 6
2560 POKE 16384+2048+2*32+J*256+
31-I,PEEK (M+I)
2570 NEXT J: NEXT I
2575 PLOT 0,95: DRAW 255,0: PLOT
 0,88: DRAW 255,0
2580 PRINT '"AREA DEL MAPA     :"
;M;"-";M+31
2590 PRINT "NOMBRE CARTUCHO  :";
2600 FOR I=D+44 TO D+53
2610 PRINT CHR$ PEEK I;
2620 NEXT I: PRINT
2630 PRINT "NOMBRE FICHERO   :";
2640 FOR I=D+14 TO D+23
2650 PRINT CHR$ PEEK I;
2660 NEXT I: PRINT '"ESPACIO LIB
RE    :";
2670 LET F=0
2680 FOR I=0 TO 255
2690 LET F=F+NOT POINT (I,90)
2700 NEXT I
2710 PRINT F/2;"KBYTES"
2720 GO TO 1500
```

Entre los detalles adicionales están el número de la unidad de cartucho, el número del cartucho, su nombre y el nombre del fichero así como el espacio libre que queda en el cartucho elegido. Asimismo, se muestra un «mapa de la cinta» que viene a ser un diagrama de barras que muestra qué áreas de la cinta están ocupadas y qué áreas están libres. La **figura 3** muestra la salida del programa con un cartucho recién estrenado. Las zonas oscuras señalan las secciones de la cinta que o bien no existen o bien están ocupadas. La zona negra de mayor tamaño que puede verse a la izquierda indica una sección de la cinta que, simplemente, no existe puesto que el sistema es capaz de trabajar con cintas de mayor longitud. La barra central señala dónde está la unión en la cinta y la de la extrema derecha muestra el primer sector (que nunca se emplea). El «mapa» de una cinta guardada en un cartucho con más tiempo de uso contendría más barras verticales que mostrarían dónde hay guardados programas o datos.

```
     NUMERO DE CORRIENTE 10

IDENTIFICADOR DE CANAL      :M
MICRODRIVE
RUTINA DE SALIDA            :8
RUTINA DE ENTRADA           :8
SALIDA DE LA ROM ADICIONAL:4568
ENTRADA DE LA ROM           :4386
UNIDAD NUMERO               :1
MAPA DE LA CINTA            :

[illegible] DEL MAPA        :23792-23823
NOMBRE CARTUCHO        :21.8.83 3
NOMBRE FICHERO         :TEST
ESPACIO LIBRE          :90Kbytes

FIGURA 3    CATALOGO DE CORRIENTES PARA CARTUCHO NUEVO
```

La principal modificación que se ha añadido al programa «Catálogo de corrientes» original está entre las líneas 1840 y 1860. Estas líneas comprueban el tipo de los canales del Interface: si son correctos, se imprimen las posiciones de memoria de las rutinas de entrada y salida de la ROM adicional. Estas líneas adicionales también son [illegible]

empleadas por las posteriores ampliaciones del programa referentes al RS 232 y al enlace a red. Las líneas 2500 a 2720 imprimen diversos detalles referentes a la corriente. El «mapa» de la cinta se imprime llevando los datos de la memoria a la pantalla mediante POKEs (líneas 2530-2575). El espacio libre se calcula contando el número de barras en blanco del «mapa» que aparece en la pantalla (2670-2710).

Rutinas de clasificación para juegos

Como aplicación que demuestre el uso de los ficheros de datos aquí van una serie de rutinas que pueden servir para elaborar una tabla de puntuaciones para un juego, tabla que se guardará en un fichero. En el programa hay dos subrutinas. La primera —que empieza en la línea 9000— (debe ejecutarse antes de empezar el juego) rellena la tabla y la segunda —iniciada en la línea 9100— actualiza la tabla cuando se consigue una puntuación superior a la más alta registrada con anterioridad.

```
8996 REM ************************
8997 REM * RUTINA CLASIFICACION *
8998 REM ************************
8999 REM LECTURA DATOS
9000 LET ns=5: DIM s(ns): DIM s$
(ns,10)
9010 CLOSE #4: OPEN #4;"m";1;"PU
NTUACION MAXIMA"
9020 FOR i=1 TO ns
9030 INPUT #4;s(i); LINE s$(i)
9040 NEXT i: CLOSE #4
9050 GO TO 9300
9098
9099 REM CREAR DE NUEVO EL FICHE
RO SI ES PUNTUACION MAXIMA
9100 LET ns=5: IF sc<s(ns) THEN
 RETURN
9110 PRINT ' FLASH 1;" NUEVA PUN
TUACION MAXIMA!!!!      "
9120 INPUT "ENTRA TU NOMBRE (MAX
.10 LETRAS)"; LINE a$
```

```
9130 FOR i=1 TO ns
9140 IF sc<=s(i) THEN  NEXT i
9150 ERASE "m";1;"PUNTUACION MAX
IMA"
9155 OPEN #4;"m";1;"PUNTUACION M
AXIMA"
9160 FOR j=1 TO ns
9170 IF j<i THEN  PRINT #4;s(j)'
s$(j)
9180 IF j=i THEN  PRINT #4;sc'a$
9190 IF j>i THEN  PRINT #4;s(j-1
)'s$(j-1)
9200 NEXT j
9210 CLOSE #4
9220 RETURN
9299 REM IMPRIMIR TABLA PUNTUACI
ONES
9300 CLS
9310 PRINT  FLASH 1; INK 7; PAPE
R 0;TAB 6;"PUNTUACIONES MAXIMAS"
;TAB 31;" "
9320 FOR i=1 TO ns
9330 PRINT  PAPER i; INK 9,,TAB
6;i;"  ";s$(i);TAB 20;s(i),,,
9340 NEXT i
9350 RETURN
9988
9989 REM CREAR NUEVO FICHERO
9990 OPEN #4;"m";1;"PUNTUACION M
AXIMA"
9992 FOR i=1 TO 5
9994 PRINT #4;0'" "
9996 NEXT i : CLOSE #4
```

La tabla se almacena en forma de diez elementos ordenados en parejas: primera puntuación, primer nombre, segunda puntuación, segundo nombre, etc. Las líneas 9900 y posteriores sólo deben ejecutarse una vez puesto que rellenan el fichero con datos nulos. Para rellenar la tabla se emplean las líneas 9000 —9050. La línea 9000 fija el número de elementos de la tabla, **ns**, e inicializa los vec-

tores s() para cada puntuación y s$() para los nombres. El fichero se cierra. Luego, se imprime la tabla a partir de la línea 9300. Una vez terminado el juego, se guarda la puntuación en la variable sc y se ejecuta la instrucción GOSUB 9100. La línea 9100 comprueba si la puntuación es más pequeña que la menor existente en la tabla. Si lo es, se ejecuta RETURN, puesto que el jugador no se ha clasificado entre los mejores. En caso de que haya conseguido clasificarse, debe entrar su nombre (línea 9120). Después se calcula qué posición le corresponde ocupar en la tabla (9130-9140). La variable i guarda esa posición. La línea 9150 borra el fichero anterior y la 9155 crea un nuevo fichero de escritura. El bucle que va de las líneas 9160 a 9200 escribe en el fichero la nueva tabla insertando el nuevo nombre y la nueva puntuación y desplazando una posición hacia abajo cualquier información que ya estuviera allí. Finalmente la línea 9210 cierra el fichero.

Capítulo 4

EL «UNIFILE» (un programa completo para gestionar una base de datos con los microdrives)

El programa contenido en este Capítulo es, en realidad, un programa completo para la gestión de una base de datos y utiliza los Microdrives. Opcionalmente, puede trabajar con una impresora (del tipo ZX o RS 232). El «Unifile» es un programa basado en buena parte en el programa «Unifile 1» desarrollado por David Lawrence y recogido en su libro «The Working Spectrum». Se ha contado con su autorización para ampliarlo aquí.

El programa está destinado a usuarios de Spectrum con 48 K de memoria. De todas formas, se dan detalles para desarrollar una versión reducida utilizable en versiones de 16 K. No se ha renumerado el programa: las líneas extraídas del programa original son idénticas.

Listado del «Unifile»

```
1000 PAPER 7: CLS : BORDER 7: IN
K 6: PAPER 0: PRINT  PAPER 2;"
 UNIFILE ";F$;TAB 31;" "
1005 LET all=0
1010 PRINT '"FUNCIONES DISPONIBL
ES:"
1020 PRINT '"      1)CREAR UN NUE
VO FICHERO"
1030 PRINT '"      2)ENTRAR INFOR
MACION"
1040 PRINT '"      3)BUSQUEDA/PRE
SENTACION/         MODIFICACION
"
```

```
1050 PRINT '"      4)STOP"
1055 PRINT '"      5)SAVE/LOAD/CA
T"
1060 PRINT ''"ENTRAR LA QUE SE D
ESEE EMPLEAR."
1070 PAUSE 0: LET Z$=INKEY$
1080 CLS
1090 IF Z$="1" THEN  GO SUB 1210
1100 IF Z$="2" THEN  GO SUB 1440
1110 IF Z$="3" THEN  GO SUB 2180
1120 IF Z$="4" THEN  GO SUB 1150
1125 IF Z$="5" THEN  GO TO 3500
1130 CLS
1140 GO TO 1000
1150 PRINT AT 10,2; INK 7; PAPER
 2;"SISTEMA DE FICHEROS CERRADO"
1160 BEEP 2,2
1180 INPUT "HA INTRODUCIDO NUEVA
 INFORMACION QUE DESEE GUARDAR(S
AVE)(S/N)";Q$: IF Q$="N" THEN  S
TOP
1190 SAVE "UNIFILE": PRINT '"REB
OBINE LA CINTA Y PULSE CUAL- QUI
ER TECLA PARA VERIFICAR": PAUSE
0: VERIFY "UNIFILE": STOP
1200 REM ***********************
1210 REM ESTRUCTURA DE LA
         ENTRADA
1220 REM ***********************
1230 PRINT  PAPER 2;"     ESTRUC
TURA DEL FICHERO      "
1235 GO SUB 4200
1240 PRINT '"CUANTOS ELEMENTOS P
OR ENTRADA? "
1250 INPUT X
1260 CLS
1270 DIM A$(X,20)
1280 PRINT  PAPER 2;"   NOMBRES
DE LOS ELEMENTOS     "
1290 FOR I=1 TO X
1300 PRINT "ELEMENTO ";I;":";
```

```
1310 GO SUB 2780
1320 PRINT Q$(2 TO )
1330 LET A$(I)=Q$
1340 NEXT I
1350 DIM B$(28000)
1360 LET B$(1 TO 4)=CHR$ 2+CHR$
0+CHR$ 2+CHR$ 255
1370 DEF FN A()=256*CODE Y$(2*S-
1)+CODE Y$(2*S)
1380 DEF FN A$()=B$(C TO C+CODE
B$(C)-1)
1390 LET P=5
1400 LET Y$=CHR$ 0+CHR$ 1+CHR$ 0
+CHR$ 3
1410 LET N=2
1420 RETURN
1430 REM ************************
1440 REM ENTRADA NORMAL(INPUT)
1450 REM ************************
1460 LET R$=""
1470 PRINT  PAPER 2;"           EN
TRADAS                 "
1480 PRINT '"INSTRUCCIONES DISPO
NIBLES:"
1490 PRINT '">ENTRAR EL ELEMENTO
 ESPECIFICADO"''">""ZZZ"" PARA A
CABAR"
1500 PRINT "*********************
***********"
1510 PRINT "TAMANO DEL FICHERO:"
;P-1;"/";LEN B$
1520 FOR I=1 TO X
1530 GO SUB 2810
1540 GO SUB 2780
1580 PRINT Q$(2 TO )
1590 IF Q$(2 TO )="ZZZ" THEN  RE
TURN
1600 LET R$=R$+Q$
1610 NEXT I
1620 CLS
1630 GO SUB 1660
```

```
1640 GO TO 1440
1650 REM ************************
1660 REM COLOCAR LOS DATOS EN EL
         FICHERO
1670 REM ************************
1680 IF P+LEN R$-1<LEN B$ THEN
GO TO 1730
1690 PRINT AT 14,10;"FICHERO LLE
NO"
1700 PRINT ''" PULSAR CUALQUIER
TECLA PARA      CONTINUAR"
1710 PAUSE 0
1720 RETURN
1730 LET POTENCIA=INT (LN (N-1)/
LN 2)
1740 LET S=2↑POTENCIA
1750 LET T$=R$(2 TO CODE R$(1))
1760 FOR K=POTENCIA-1 TO 0 STEP -1
1770 LET C=FN A()
1780 LET U$=FN A$()(2 TO )
1790 LET S=S+(2↑K)*(T$>U$)-(2↑K)
*(T$<U$)
1810 IF S>N-1 THEN  LET S=N-1
1820 IF S<2 THEN  LET S=2
1830 NEXT K
1840 LET C=FN A()
1850 LET U$=FN A$()(2 TO )
1860 IF T$<U$ THEN  LET S=S-1
1870 LET B$(P TO P+LEN R$-1)=R$
1880 LET N=N+1
1890 LET Y$=Y$(1 TO 2*S)+CHR$ IN
T (P/256)+CHR$ (P-256*INT (P/256
))+Y$(2*(S+1)-1 TO )
1900 LET P=P+LEN R$
1910 RETURN
1920 REM ************************
1930 REM CAMBIAR ENTRADA
1940 REM ************************
1950 LET S=S-1
1960 LET C=FN A()
1970 LET R$=""
```

```
1980 PRINT "ENTRADA ";S-1;":-"
1990 FOR I=1 TO X
2000 GO SUB 2810
2010 GO SUB 2830
2020 PRINT AT 17,0; PAPER 2;"
      MODIFICAR                 "
2030 PRINT "INSTRUCCIONES DISPON
IBLES:"
2040 PRINT ">""ENTER"" NO CAMBIA
 EL ELEMENTO"'">""ZZZ"" BORRA TO
TALMENTE LA ENTRADA"'">ENTRAR NU
EVO ELEMENTO"
2050 GO SUB 2780
2060 IF LEN Q$=1 THEN  LET R$=R$
+B$(C TO C+CODE B$(C)-1)
2070 LET C=C+CODE B$(C)
2080 CLS
2090 IF LEN Q$=1 THEN  GO TO 212
0
2100 IF Q$(2 TO )="ZZZ" THEN  GO
 TO 2130
2110 LET R$=R$+Q$
2120 NEXT I
2130 GO SUB 3130
2140 IF Q$(2 TO )="ZZZ" THEN  RE
TURN
2150 GO SUB 1660
2160 RETURN
2170 REM ************************
2180 REM BUSQUEDA
2190 REM ************************
2200 LET S=2
2205 LET ST=2
2210 PRINT  PAPER 2;"           B
USQUEDA                "
2220 PRINT ''"INSTRUCCIONES DISP
ONIBLES:"
2230 PRINT ">ENTRAR ELEMENTO PAR
A BUSQUEDA   NORMAL"'">PRECEDERL
O CON ""SSS"" PARA"'" BUSQUEDA E
SPECIAL"'">O CON ""III"" PARA BU
```

```
SCAR"'" SEGUN PRIMER CARACTER DE
 LA      ENTRADA"'">ENTER"" PARA
EL PRIMER ELEMENTO DEL FICHERO"
2235 PRINT ">""PPP"" PARA ";"IMP
RESORA " AND ST=2;"PANTALLA" AND
 ST=3;" SALIDA"
2240 PRINT "********************
************"
2250 PRINT ''"ENTRAR EL ELEMENTO
 A BUSCAR:"
2260 GO SUB 2780
2265 IF Q$=CHR$ 4+"PPP" THEN  LE
T ST=5-ST: CLS : GO TO 2210
2270 PRINT Q$(2 TO )
2280 LET S$=Q$
2290 IF LEN S$=1 THEN  GO TO 251
0
2300 LET C=FN A()
2310 IF LEN S$<5  THEN  GO TO 24
30
2320 IF S$(2 TO 4)<>"III" THEN
GO TO 2390
2330 FOR I=S TO N
2340 LET S=I
2350 LET C=FN A()
2360 IF B$(C+1)=S$(5) THEN  GO T
O 2510
2370 NEXT I
2380 RETURN
2390 IF S$(2 TO 4)<>"SSS" THEN
GO TO 2430
2400 GO SUB 2920
2410 IF C4=1 THEN  GO TO 2510
2420 RETURN
2430 FOR I=1 TO X
2440 IF FN A$()=S$ THEN  GO TO 2
510
2450 IF FN A$()=CHR$ 2+CHR$ 255
THEN  RETURN
2460 LET C=C+CODE B$(C)
2470 NEXT I
```

```
2480 LET S=S+1
2490 LET C=FN A()
2500 GO TO 2430
2510 LET C=FN A()
2520 LET C4=0
2530 IF FN A$()=CHR$ 2+CHR$ 255
THEN  RETURN
2540 CLS
2545 IF ST<>2 THEN  GO TO 4300
2550 PRINT "ENTRADA ";S-1;":-"
2560 GO SUB 2850
2570 LET S=S+1
2580 PRINT AT 16,0; PAPER 2;"
     BUSQUEDA                 "
2590 PRINT "INSTRUCCIONES DISPON
IBLES:"
2600 PRINT ">""ENTER"" PARA PRES
ENTAR SI-.      GUIENTE ELEMENTO"
'">""ZZZ""PARA ACABAR LA FUNCION
"'">""AAA"" PARA MODIFICAR"'">""
CCC"" PARA CONTINUAR LA BUSQUEDA
"
2610 INPUT P$
2620 CLS
2630 IF P$="CCC" THEN  GO TO 230
0
2640 IF P$="" THEN  GO TO 2510
2650 IF P$<>"AAA" THEN  GO TO 27
10
2660 LET C=FN A()
2670 CLS
2680 GO SUB 1930
2710 IF P$="ZZZ" THEN  RETURN
2720 IF P$="AAA" THEN  RETURN
2730 CLS
2740 GO TO 2260
2750 REM ***********************
2760 REM RUTINAS ENCARGADAS DE
         LAS FUNCIONES
2770 REM ***********************
2780 INPUT q$
```

```
2790 LET Q$=CHR$ (LEN Q$+1)+Q$
2800 RETURN
2810 PRINT A$(I,2 TO CODE A$(I,1
));":";
2820 RETURN
2830 PRINT FN A$()(2 TO )
2840 RETURN
2850 FOR I=1 TO X
2860 GO SUB 2810
2870 GO SUB 2830
2880 LET C=C+CODE B$(C)
2890 NEXT I
2900 RETURN
2910 REM ************************
2920 REM BUSQUEDA ESPECIAL
2930 REM ************************
2940 LET C4=0
2950 FOR H=S TO N-1
2960 LET S=H
2970 LET C=FN A()
2980 LET C1=C
2990 FOR I=1 TO X
3000 LET C1=C1+CODE B$(C1)
3010 NEXT I
3020 FOR J=C+1 TO C1-LEN S$+5
3030 IF B$(J TO J+LEN S$-5)<>S$(
5 TO ) THEN  GO TO 3060
3040 LET C4=1
3050 RETURN
3060 NEXT J
3070 NEXT H
3080 LET C4=0
3090 RETURN
3100 REM ************************
3110 REM FICHERO TELESCOPICO
3120 REM ************************
3130 LET C=FN A()
3140 LET SHIFT=1000
3150 LET C1=C
3160 LET C3=C
3170 FOR I=1 TO X
```

```
3180 LET C1=C1+CODE B$(C1)
3190 NEXT I
3200 LET C2=C1-C
3210 FOR I=C1 TO LEN B$-1 STEP S
HIFT
3220 IF LEN B$-I+1<SHIFT THEN  L
ET SHIFT=LEN B$-I+1
3230 LET S$=B$(I TO I+SHIFT-1)
3240 LET B$(C TO C+SHIFT-1)=S$
3250 LET C=C+SHIFT
3260 NEXT I
3270 LET Y$=Y$(1 TO 2*(S-1))+Y$(
2*(S+1)-1 TO )
3280 FOR I=1 TO N-1
3290 LET S=I
3300 LET C=FN A()
3310 IF C<=C3 THEN  GO TO 3350
3320 LET C=C-C2
3330 LET Y$(2*I-1)=CHR$ INT (C/2
56)
3340 LET Y$(2*I)=CHR$ (C-256*INT
 (C/256))
3350 NEXT I
3360 LET P=P-C2
3370 LET N=N-1
3380 RETURN
3390 FOR I=1 TO 20: PRINT CODE Y
$(I): NEXT I
3500 CLS
3510 PRINT "UNIFILE - OPERACIONE
S CON EL MI CRODRIVE"
3520 PRINT '"  1) ALMACENAR (SAV
E) "; INVERSE 1;F$
3530 PRINT '"  2)CARGAR (LOAD) N
UEVOS DATOS"
3540 PRINT '"  3) CATALOGO (CAT)
 DE UN            CARTUCHO"
3545 PRINT '"  5) RETORNO (RETUR
N) AL MENU         PRINCIPAL"
3550 PAUSE 0: LET Z$=INKEY$
3560 IF Z$="1" THEN  GO SUB 3900
```

```
3570 IF Z$="2" THEN  GO TO 3800
3580 IF Z$="3" THEN  GO SUB 3700
3590 IF Z$="4" THEN  GO TO 1000
3600 GO TO 3500
3670 REM ********************
3680 REM RUTINA DE CATALOGO
3690 REM ********************
3700 CLS : PRINT "CATALOGO DEL C
ARTUCHO:"
3710 INPUT "NUMERO DE UNIDAD DE
CARTUCHO?   (0 PARA FINALIZAR)";
D
3720 IF D=0 OR D>8 THEN  RETURN
3730 CAT D
3740 PRINT #0;AT 0,0; FLASH 1;"P
ULSAR CUALQUIER TECLA PARA     S
EGUIR";
3750 PAUSE 0
3760 RETURN
3770 REM **********************
3780 REM RUTINA DE CARGA (LOAD)
3790 REM **********************
3800 PRINT '' FLASH 1;"ATENCION:
LA CARGA DE NUEVOS DA  TOS, BORR
A LOS ANTERIORES"
3805 PRINT '"PULSAR C PARA CONTI
NUAR, O CUAL QUIER OTRA TECLA PA
RA TERMINAR."
3810 PAUSE 0: LET Z$=INKEY$: IF
Z$<>"C" AND Z$<>"c" THEN  GO TO
3500
3820 GO SUB 4200
3830 INPUT "NUMERO DE UNIDAD (1-
8)? ";D
3840 IF D<0 OR D>8 THEN  GO TO 3
830
3850 LOAD *"M";D;F$+CHR$ 128
3860 GO TO 1000
3870 REM **********************
3880 REM RUTINA DE ALMACENAMIEN
         TO (SAVE)
3890 REM **********************
```

```
3900 PRINT ''"(NOMBRE EXISTENTE=
";F$;")"
3905 GO SUB 4200
3910 INPUT "NUMERO DE UNIDAD? ";
D
3920 IF D<0 OR D>8 THEN  RETURN
3930 CLS
3940 PRINT "   UNIFILE - ";F$
3950 PRINT AT 15,0;"QUIERE BORRA
R EL FICHERO ";F$'" DE LA UNIDAD
 ";D;"?   (S/N)"
3960 INPUT  LINE z$
3965 PRINT AT 15,0;,,,,
3970 IF Z$<>"S" AND Z$<>"s" THEN
  GO TO 4000
3980 ERASE "M";D;F$+CHR$ 128
3990 PRINT '"FICHERO BORRADO"
4000 SAVE *"M";D;F$+CHR$ 128 LIN
E 1000
4010 PRINT "DATOS ALMACENADOS -
SE ESTAN    VERIFICANDO"
4020 VERIFY *"M";D;F$+CHR$ 128 L
INE 1000
4030 RETURN
4170 REM **********************
4180 REM ENTRADA DEL NOMBRE DEL
         FICHERO
4190 REM **********************
4200 INPUT "NOMBRE DEL FICHERO?
"; LINE F$
4210 IF LEN F$<10 AND LEN F$>0 T
HEN  RETURN
4220 INPUT "(LONGITUD MAX. 9) ";
 LINE F$
4230 GO TO 4210
4270 REM ********************
4280 REM SALIDA POR IMPRESORA
4290 REM ********************
4300 LPRINT "ENTRADA ";S-1;":-"
4310 FOR I=1 TO X
```

```
4320 LPRINT A$(I,2 TO CODE A$(I,
1));":";
4330 LPRINT FN A$()(2 TO )
4340 LET C=C+CODE B$(C)
4350 NEXT I
4360 GO TO 2570
```

Una vez se haya introducido el programa, hay que incorporar las siguientes instrucciones:

```
CLEAR
LET F$=""
```

y hay que guardar el programa (mediante SAVE) con LINE 1000. Para iniciar su ejecución hágase **GO TO 1.** (Nunca ha de emplearse la instrucción RUN: provoca el borrado de los datos).

Cada fichero de datos puede contener hasta 28.000 caracteres organizados en elementos definidos en el momento de inicializar el fichero. Por ejemplo, éste podría ser una agenda con tres elementos por entrada: nombre, dirección y número de teléfono.

La **opción 1** permite crear un nuevo fichero y pide al usuario que le asigne un nombre.

La **opción 2** permite añadir información al fichero.

La **opción 3** es la más útil. Permite realizar una búsqueda sobre todo el fichero y, luego, escribe en la pantalla las entradas apropiadas o bien las imprime en la impresora. La opción de «búsqueda especial» intentará encontrar un elemento dentro del fichero que tenga alguna de sus tres partes iguales a la cadena que se especifique. No se trata de una operación excesivamente rápida. Por ejemplo, si se entra **SSS-MADRID** y se trabaja sobre el fichero de direcciones, se imprimirá el nombre de todas aquellas personas de la agenda que vivan en Madrid o cuyo apellido sea Madrid. Una operación más rápida es la búsqueda normal que tan sólo inspecciona el primer carácter de cada entrada.

La **opción 4** permite que el programa y los datos se graben en cassette para disponer de copias de seguridad.

La **opción 5** permite almacenar los datos existentes en un momento dado o bien cargar otros nuevos o catalogar un cartucho.

Los ficheros se guardan en cartucho con un CHR$ 128 al final para poder distinguirlos de los ficheros normales de programas. Este carácter adicional no queda recogido en el CATálogo ya que se imprime como un espacio.

No voy a explicar el funcionamiento del programa: David Lawrence lo hace muy bien en su libro. Explicaré, en cambio, la subrutina de la impresora con el fin de que los usuarios puedan modificarla para que se adapte mejor a las especificaciones de sus ficheros. La rutina va de las líneas 4300 a la 4350. **S-1** es la posición que ocupa la entrada dentro del fichero y **X** es el número de elementos por entrada. La línea 4320 imprime el título de cada entrada y la 4330 los datos en sí. La línea 4340 incrementa un puntero. Si, por ejemplo, se hubiera creado un fichero que sirviera como agenda de direcciones tal y como se ha indicado anteriormente y se quisiera imprimir las entradas en forma de etiqueta de dirección para correspondencia, podría hacerse lo siguiente:

```
4300 LPRINT "NUM. REF.";S-1
4310 LPRINT FN A$()(2 TO ): REM
EL NOMBRE
4320 LET C=C+CODE B$(C)
4330 LPRINT FN A$()(2 TO ): REM
LA DIRECCION
4340 LPRINT
4350 LET C=C+CODE B$(C)
4360 LPRINT "TELEF.:";FN A$()(2
TO ): REM EL NUMERO
4370 LET C=C+CODE B$(C)
4380 GO TO 2570
```

Nuevas posibilidades

Se pueden añadir otras posibilidades a las que ya se ofrecen. Una de ellas podría ser la de disponer de una operación CAT que sólo catalogara los programas del «Unifile». Para ello habría que recurrir a la rutina «Corriente 14-z$» y examinar el contenido de Z$ para ver si aparece algún CHR$ 128. Otra mejora podría ser la de traducir las líneas 3130-3380 a código máquina con lo cual el borrado y la modificación de elementos se aceleraría notablemente.

Versión para 16 K

Se puede utilizar una versión restringida del programa «Unifile» en sistemas con 16 K de memoria. Hay que introducir las siguientes modificaciones:

Se deben borrar todas las instrucciones REM, las líneas 3540, 3580 y 3700-3760.

Hay que cambiar las líneas:

```
1350 DIM B$(1000)
3140 LET SHIFT=100
3375 LET S$=""
```

Capítulo 5
PROTECCION DE PROGRAMAS

Cuando uno ha escrito un buen softward y quiere venderlo no tiene ningún interés en que cualquiera pueda copiar su obra maestra y repartirla entre sus amigos. Aunque no hay programa para el que se pueda garantizar una seguridad del 100 % contra las copias piratas, existen muchas técnicas sencillas que permiten dificultar el proceso de copia en gran manera. Explicaremos aquí un par de posibilidades ya contenidas en el propio ordenador y otras técnicas más astutas. Es necesario subrayar que no hay métodos imbatibles. Con tiempo y conocimientos suficientes un determinado programador en lenguaje máquina puede ser capaz de vencer las barreras de protección puestas para dificultar el acceso a un programa.

Auto-ejecución

La forma más sencilla de dificultar la copia de programas es hacer que sean capaces por sí mismos de iniciar su ejecución. Para ello se recurre a la función SAVE*... LINE. No puede ser fundida (MERGE). Por tanto, cuando se carga, provoca el auto-inicio. Si a un programa se le da el nombre de «run» (en minúsculas) entonces, si después de conectar el sistema o hacer NEW se teclea RUN, el programa se cargará y automáticamente empezará a ejecutarse (si se guardó mediante SAVE y LINE).

También resulta útil hacer que los nombres de los programas empiecen por el carácter CHR$ 0. De esta forma no aparecerán en el CATálogo del cartucho. Si se hace que el programa «run» cargue el programa «invisible», el posible pirata de software será incapaz de saber siquiera el nombre del programa principal. Otro sistema consiste en hacer que el nombre de un programa esté compuesto únicamente por espacios en blanco: la mayoría de los usuarios ni se darán cuenta de que está en el catálogo.

POKEs «fatales»

Algo que hay que impedir a toda costa es que el usuario pueda entrar en el programa una vez está en ejecución o mientras se está realizando un LOAD. En el caso explicado anteriormente esto último le permitiría ver el nombre del fichero «invisible» mediante el examen de su listado. Una manera de impedir esto que, aunque no es muy elegante, funciona perfectamente, consiste en hacer que la primera línea sea:

```
POKE 23659,0
```

Lo que hace esta línea es decirle al Spectrum que no hay más líneas en la parte inferior de la pantalla. En caso de que ocurra algún error (como el de **Break into program** que se produciría si alguien entrara en el programa) hará que todo el sistema quede «colgado» al no haber espacio suficiente para escribir el mensaje de error. El usuario tendrá que desconectar el ordenador para poner en marcha de nuevo el sistema. De esta forma se impide que examine o copie el listado del programa.

Hay que ir con tiento al emplear este POKE particular pues una instrucción CLS o cualquier intento de escribir en las dos líneas inferiores (por ejemplo con INPUT) resultará fatal para el ordenador. El POKE es únicamente útil cuando se emplean programas en código máquina y en especial mientras dura su proceso de carga. Para programas en BASIC resulta una solución demasiado limitada. Un método alternativo es el que utiliza la línea siguiente:

```
LET p=PEEK 23613+256*PEEK 23614:
POKE p,0:POKE p+1 ,0
```

que será fatal para el sistema en cuanto ocurra un error pero que, en cambio, permite escribir en la parte inferior de la pantalla.

On error GOTO

Muchos ordenadores tienen una función «On error goto» que provoca un salto a una determinada línea cuando se produce un error.

Desgraciadamente, el Spectrum no dispone de ella. La siguiente rutina en código máquina realiza las mismas acciones que esa función:

Listado de «On error goto»

```
8794 REM ********************
8795 REM *  ON ERROR GOTO   *
8796 REM ********************
8797 REM err=DIRECCION DE INICIO
, linea=SALTO DE ERROR
8798 REM RECOMENDADO:64780/32010
8800 RESTORE 8900: LET c=0
8810 FOR i=err TO err+170
8820 READ a: LET c=c+a
8830 IF a<256 THEN  POKE i,a
8840 NEXT i
8850 IF c<>17439 THEN  PRINT "ER
ROR EN SUMA DE CONTROL": STOP
8860 POKE err+149,linea-256*INT
(linea/256)
8870 POKE err+150,INT (linea/256
)
8875 RETURN
8880 RANDOMIZE USR err
8885 POKE 23734,0: RETURN
8890 RANDOMIZE USR err
8895 POKE 23734,4: RETURN
8900 DATA 33,17,0,9,235,42,61,92
8905 DATA 115,35,114,207,49,1,0,
0
8910 DATA 201,34,201,92,42,61,92
,17
8915 DATA 3,19,213,205,176,22,25
3,203
8920 DATA 55,174,205,110,13,33,1
85,23
8925 DATA 34,237,92,58,58,92,50,
211
8930 DATA 92,245,207,50,33,129,0
,237
8935 DATA 91,201,92,241,167,237,
```

```
82,32
8940 DATA 20,253,203,124,70,32,1
4,254
8945 DATA 7,40,10,62,100,50,211,
92
8950 DATA 62,63,215,24,21,60,71,
254
8955 DATA 10,56,2,198,7,205,239,
21
8960 DATA 62,32,215,120,17,145,1
9,205
8965 DATA 10,12,175,17,54,21,205
,10
8970 DATA 12,237,75,69,92,237,67
,201
8975 DATA 92,205,27,26,62,58,215
,253
8980 DATA 78,13,6,0,205,27,26,33
8985 DATA 59,92,203,174,251,203,
110,40
8990 DATA 252,33,66,92,17,260,27
0,115
8994 DATA 35,114,35,54,1,253,54,
0
8995 DATA 255,253,54,124,0,205,1
10,13
8996 DATA 195,125,27
```

La rutina tiene una longitud de 171 bytes y es totalmente independiente de la posición de memoria en que se cargue. Antes de la instrucción GO SUB 8800 que inicia la ejecución, hay que dar a **err** el valor de la posición de memoria requerida y a **line** el valor de la línea de error. Con operaciones normales (aquellas que no hagan referencia al Interface) se hace GO SUB 8800 para activar la rutina. Cuando tiene lugar un error se imprime el mensaje correspondiente en el lugar normal. Entonces se produce una pausa en espera de que se pulse una tecla. Cuando se ha hecho esto, se borra el mensaje y se salta a la línea correspondiente. Se produce una desactivación cuando se origina un error. El número de la línea donde se ha producido el error puede conocerse haciendo:

```
PEEK 23753+256*PEEK 23754
```

y el número del error haciendo:

```
PEEK 23763
```

Para poner en marcha la rutina con las operaciones que hagan referencia al Interface se debe hacer GO SUB 8890. Téngase en cuenta que si se produce un error del Interface (por ejemplo, un error que no tiene mensaje de error) no aparecerá el debido mensaje de error sino que se imprimirá un signo de interrogación y habrá el valor 100 en la posición de memoria 23763.

Después de cada operación relacionada con el Interface la rutina queda inhabilitada (es decir, no se permite que entre en ejecución) como resultado de su propio funcionamiento.

Para inhabilitar la rutina hay que hacer:

```
LET p=PEEK 23613+256*PEEK 23614:
 POKE p,3: POKE p+1,19: POKE 237
34,0
```

Esta rutina no es incompatible con la rutina «On EOF GOTO» del Capítulo anterior. Para saber si se ha producido un error de fin de fichero basta con comprobar si en la posición 23763 hay el valor 7 (o, dicho con otras palabras, la operación PEEK 23763 dará por resultado un 7).

Cuando depure sus programas tenga en cuenta los bucles infinitos. Si la línea de error fuera la 100 y tuviera esta forma:

```
1000 GO SUB 8880
```

entonces es muy posible que no se pudiera salir nunca del programa. Inclúyase siempre en los programas una opción que permita realizar el POKE necesario para conseguir la inhabilitación.

Para incluir esa rutina en un programa comercial lo mejor es añadir la siguiente línea:

```
8855 POKE err+23,94: POKE err+24
,35: POKE err+25,86
```

Estos POKEs modifican la rutina de tal manera que no se cancele cuando haya un error.

Si usted quiere insertar en un programa comercial esta rutina o cualquier otra rutina en código máquina recogida en este libro tendrá que obtener primero mi autorización puesto que las tengo registradas.

Como hemos visto, con un poco de código máquina y unos cuantos POKEs se puede obtener una protección efectiva de cualquier programa. Desgraciadamente la proposición inversa también es cierta: con algo de código máquina se puede «romper» un programa deshaciendo su sistema de protección. Para vencer las barreras de protección es necesario que el programador sepa qué métodos de protección se han utilizado. Esto explica por qué resulta relativamente sencillo para un programador en código máquina obtener un catálogo de ficheros «invisibles» o fusionar programas que se ponen en marcha por sí mismos. Es un desgraciado efecto secundario de la extensa disponibilidad de detalles técnicos de los productos Sinclair.

Capítulo 6

EMPLEO DEL INTERFAZ RS232

Otra característica del Interface 1 es su capacidad para transmitir y recibir datos a través del RS232. El RS232 es (casi) una norma universal para transmisión en serie. Hay una gran cantidad de periféricos y ordenadores que pueden comunicarse a través de este interfaz. El Spectrum emplea al RS232 principalmente para controlar las impresoras «serias» de cierta anchura para producir listados y salidas de programas.

La velocidad a la que se envían o reciben datos se mide en baudios. Un baudio equivale aproximadamente a diez veces el número de bytes enviados en un segundo. Hay nueve niveles de transmisión normalizados que corresponden a las velocidades de 50, 110, 300, 600, 1200, 2400, 4800, 9600 y 19200 baudios. El Spectrum puede trabajar con todos ellos. Los teletipos acostumbran a trabajar a velocidades bajas, de 110 ó 300 baudios. Los terminales de tubos de rayos catódicos (CRT) trabajan a una velocidad de 2400 baudios o más.

De la misma manera que los diferentes dispositivos trabajan con distintas velocidades también lo hacen con formatos de datos diferentes. En el Spectrum el formato es único y fijo:

— No hay bit de paridad.
— Ocho bits de datos.
— Un bit de parada (Stop bit).

Para emplear el RS232 en el Spectrum se utilizan dos identificadores de canales más: el «t» y el «b». Estas corrientes se diferencian en su funcionamiento por la forma en que tratan los caracteres individuales.

El canal de texto «t»

La utilización más idónea del canal «t» es el control de impresoras ya que efectúa ciertas operaciones con cada carácter. Teniendo en cuenta el conjunto de caracteres del Apéndice A del manual del Spectrum, el canal «t» imprime junto con cada código de carácter lo siguiente:

0—12	(los códigos de control) se ignoran
13	(newline) envía un retorno de carro (13) y luego avance de línea (11)
14—31	(códigos de control) se ignoran
32—127	(códigos ASCII) se envían sin ninguna modificación
128—164	(códigos gráficos) se envían como "?"
165—255	(palabras especiales) se descomponen en los caracteres individuales.

La conversión de caracteres que tiene lugar resulta ideal para transmitir listados y los resultados de programas. Sin embargo, muchas impresoras utilizan los caracteres anteriores al 32 para conseguir efectos especiales. Las impresoras Epson, por ejemplo, emplean el CHR$ 14 para imprimir textos ampliados. Sin embargo, ese carácter no se puede enviar por un canal «t». Además la función TAB (extremadamente útil) también es ignorada, lo que hace que sea más difícil obtener salidas de programa claras y ordenadas. La solución puede ser emplear el canal «b».

El canal binario «b»

El canal binario es particularmente idóneo para comunicar con dispositivos que no sean impresoras ya que no realiza ninguna conversión de caracteres. Cada carácter se transmite sin modificación alguna. Por tanto, este canal es ideal para enviar datos y programas. También resulta apropiado para enviar códigos de control a una impresora (ya que estos códigos no se pueden transmitir a través del canal «t»).

Antes de que el Spectrum pueda empezar a enviar o recibir da-

tos a través del RS232 tiene que conocer la velocidad en baudios del periférico con el que va a establecer la comunicación. Esto se consigue con la sentencia FORMAT «b» a la que debe seguir la velocidad en baudios. Por ejemplo, para una velocidad de 110 baudios hay que hacer:

```
FORMAT "b";110
```

También se podría hacer FORMAT "t"; 110. Tendría el mismo efecto. Si se escribe una velocidad distinta a las que reconoce el Spectrum no hay problema: no se produce ningún error. El Spectrum empleará la velocidad siguiente (la superior más cercana a la velocidad que se haya indicado). Para conocer qué velocidad se está empleando se puede hacer:

```
PRINT INT (3500000/((PEEK 2
3747+256*PEEK 23748+2)*26))
```

Trabajando a velocidades más altas se puede sufrir una pérdida de precisión cuantificable en algunos baudios. Si la velocidad no se especifica, el sistema no lo advertirá y trabajará con velocidades raras o realizará transmisiones defectuosas. Inicialmente, la velocidad que se toma es la de 9600 baudios.

Control de una impresora vía RS232

Antes de conectar por primera vez una impresora a través del RS232 al Spectrum habría que adecuarla para que se ajustara al formato de datos que emplea el ordenador. Normalmente, esto se consigue cambiando algunos contactos en el interior de la impresora. Sólo es necesario realizar esta operación una vez. Hay que ajustar la impresora a la velocidad más alta que permita (anótela para no olvidarla) y el formato de datos debe fijarse en: ningún bit de paridad, 8 bits de datos y uno de parada. Si la impresora ofrece la opción de avance de línea automático después del retorno de carro (en la jerga informática LF automático después de CR) hay que inhabilitarla puesto que el Spectrum es capaz de generar los «LF» necesarios.

Una vez se han fijado todos estos parámetros, se puede pasar a conectar la impresora mediante un cable adecuado al zócalo de tipo D de 9 patillas del Interface.

Cuando se controla una impresora con Interfaz del tipo RS232 resulta aconsejable tener los canales «t» y «b» abiertos simultáneamente (el primero para texto y listados y el segundo para códigos de control). Es conveniente dedicar la corriente 3 a los canales «t» de la impresora para poder emplear las instrucciones LLIST y LPRINT de programas ya existentes, por ejemplo:

```
10 FORMAT "t";1200:REM VELOCIDAD
 EN BAUDIOS
20 OPEN #3;"t":REM TEXTO EN LA C
ORRIENTE 3
30 OPEN #15;"b":REM BINARIO EN L
A CORRIENTE 15
40 LLIST :REM LISTADOS EN EL CAN
AL "t"
50 LPRINT "TAMANO NORMAL"
60 PRINT #15;CHR$ 14;#3;"ANCHURA
 DOBLE"
70 CLOSE #3: CLOSE #15
```

En este ejemplo, la corriente 15 se utiliza como un fichero «b» para transmisión de códigos de control y la corriente 3 trabaja como canal «t» para textos y listados.

Cuando se transmite a través del RS232, el borde de la pantalla toma el color negro si el dispositivo que recibe la información (en este caso la impresora) está «ocupado» y, por tanto, es incapaz de aceptar los datos que se le están enviando. Para enviarle los datos, el Spectrum esperará a que el dispositivo deje de estar ocupado o bien a que se pulse BREAK. Cuando cualquier corriente relacionada con el RS232 está cerrada, se envía un carácter 13 para vaciar las memorias intermedias («buffers») correspondientes a la o las impresoras y otros dispositivos. El carácter 13 no se envía cuando se emplea el comando CLEAR #.

No es posible tener conectado al mismo tiempo más de un dispositivo que emplee el RS232 aunque se pueden tener varias corrien-

tes conectadas al mismo canal. Tampoco es posible tener diferentes velocidades en canales o corrientes distintas.

El principal problema que se encuentra al controlar impresoras a través del RS232 es la falta del comando TAB. La siguiente rutina en código máquina abre la corriente 3 como un canal «t» pero con una ligera modificación. La rutina cuenta el número de caracteres que se envían y se hace posible la implementación plena de la función TAB. Uno se pregunta por qué esta mejora tan sencilla no se incorporó al equipo original. Esta misma rutina corrige también el error de software que, de forma incorrecta, produce dobles espacios en los listados (por ejemplo THEN PRINT). Se ha escrito el código para que esté en el área de memoria correspondiente a la memoria intermedia (buffer) de la impresora ZX, área que no se emplea cuando se redefine la corriente 3. Hay que tener en cuenta que la corriente 3 no puede asignarse de nuevo a la impresora ZX ni con una sentencia CLOSE # 3 ni con una CLEAR #. La corriente 3 no transmite los retornos de carro.

Rutina TAB

```
8997 REM ********************
8998 REM * TAB CON EL RS232 *
8999 REM ********************
9000 RESTORE 9100: LET c=0
9010 FOR i=23296 TO 23467
9020 READ a: LET c=c+a
9030 POKE i,a
9040 NEXT i
9050 POKE 23540,80: REM ANCHURA
DE LA IMPRESORA
9055 IF c<>19420 THEN  PRINT "ER
ROR EN SUMA DE CONTROL": STOP
9060 RANDOMIZE USR 23296
9070 RETURN
9100 DATA 42,79,92,1,15,0,9,17
9110 DATA 23,91,115,35,114,1,0,0
9120 DATA 33,245,91,112,35,112,2
01,254
9130 DATA 32,48,93,254,13,32,26,
33
```

```
9140 DATA 246,91,203,70,203,134,
192,33
9145 DATA 246,91,203,134,43,54,0
,62
9150 DATA 13,205,169,91,62,10,19
5,169
9160 DATA 91,254,23,63,192,17,71
,91
9170 DATA 42,81,92,115,35,114,20
1,50
9180 DATA 15,92,17,79,91,24,241,
17
9190 DATA 23,91,205,64,91,58,15,
92
9200 DATA 87,33,244,91,150,210,1
08,4
9210 DATA 35,122,150,213,220,39,
91,209
9220 DATA 122,253,150,187,200,71
,62,32
9230 DATA 197,217,215,217,193,16
,247,201
9240 DATA 254,165,56,5,214,165,1
95,16
9250 DATA 12,253,203,188,134,33,
59,92
9260 DATA 203,134,254,32,32,2,20
3,198
9270 DATA 254,128,56,2,62,63,205
,169
9280 DATA 91,33,245,91,52,126,43
,190
9290 DATA 192,205,39,91,253,203,
188,198
9300 DATA 201,207,30,201
```

Copias del contenido de la pantalla

Otra interesante posibilidad es la de utilizar una impresora ancha para obtener una copia en papel del contenido de la pantalla.

Esta posibilidad, en su forma más sencilla, se logra con las siguientes líneas de programa que funcionan con cualquier impresora que tenga, como mínimo, 32 caracteres por línea. El programa lee carácter a carácter el contenido de la pantalla y lo envía a la impresora.

```
100 FORMAT "T";BAUD:REM VALOR AD
ECUADO
110 OPEN #4;"T"
120 FOR Y=0 TO 21
130 FOR X=0 TO 31
140 LET A$=SCREEN$(Y,X)
150 IF A$= ""THEN LET A$=""
160 PRINT #4;A$;
170 NEXT X
180 PRINT #4
190 NEXT Y
```

El programa explora una a una las posiciones que ocupan los caracteres mediante la instrucción SCREEN$. En caso de que no se trate de un carácter reconocible, se imprimirá un espacio en la impresora (línea 150). En caso contrario el carácter se imprimirá normalmente. Después de cada línea de 32 caracteres se transmite un «newline» (línea 180). Aunque resulta un método un tanto tosco, la rutina no deja de ser por ello útil (en especial porque es independiente del tipo de impresora con la que se trabaje).

En caso de que se desee obtener una copia en papel de una representación gráfica en alta resolución o de un listado que contenga caracteres en representación inversa y caracteres gráficos, ¿qué se debe hacer? Si se dispone de una impresora adecuada se pueden obtener verdaderas copias como ocurre al utilizar el comando COPY de la impresora ZX. Sin embargo, en BASIC este proceso es lento aunque los resultados valgan la pena. Incluyo aquí dos versiones: una para la gama de impresoras Epson y otra para la Seikosha GP100.

Copia de alta resolución con Epson

```
1000 FORMAT "b";1200:REM CAMBIAR
PARA AJUSTARSE AL 'INTERFACE'
```

```
1010 OPEN #3;"b"
1020 LPRINT CHR$ 27;"A";CHR$ 8;
1030 FOR y=175 TO 0 STEP -8
1040 LPRINT CHR$ 27;"K";CHR$ 0;C
HR$ 1;
1050 FOR x=0 TO 255
1060 LPRINT CHR$ (128*POINT (x,y
)+64*POINT (x,y-1)+32*POINT (x,y
-2)+16*POINT (x,y-3)+8*POINT (x,
y-4)+4*POINT (x,y-5)+2*POINT (x,
y-6)+POINT (x,y-7));
1070 NEXT x
1080 LPRINT CHR$ 13;CHR$ 10;
1090 NEXT y
1100 LPRINT CHR$ 27;"A";CHR$ 12
```

Copia de alta resolución con Seikosha

```
1000 FORMAT "b";1200:REM CAMBIAR
PARA AJUSTARSE AL 'INTERFACE'
1010 OPEN #3;"b"
1020 LPRINT CHR$ 18
1030 FOR y=174 TO 0 STEP -7
1050 FOR x=0 TO 255
1060 LPRINT CHR$ (POINT (x,y)+2*
POINT (x,y-1)+4*POINT (x,y-2)+8*
POINT (x,y-3)+16*POINT (x,y-4)+3
2*POINT (x,y-5)+64*POINT (x,y-6)
+128);
1070 NEXT x
1080 LPRINT CHR$ 13;CHR$ 10;
1090 NEXT y
1100 LPRINT CHR$ 30
```

Ambas rutinas se basan en el mismo principio de funcionamiento. Recurren a la función POINT para calcular un número binario que se envía a la impresora. Obsérvese que se utiliza un canal «b» puesto que no debe producirse ningún tipo de conversión de caracteres. La **figura 4** muestra un ejemplo del resultado obtenido con la rutina para impresora Epson.

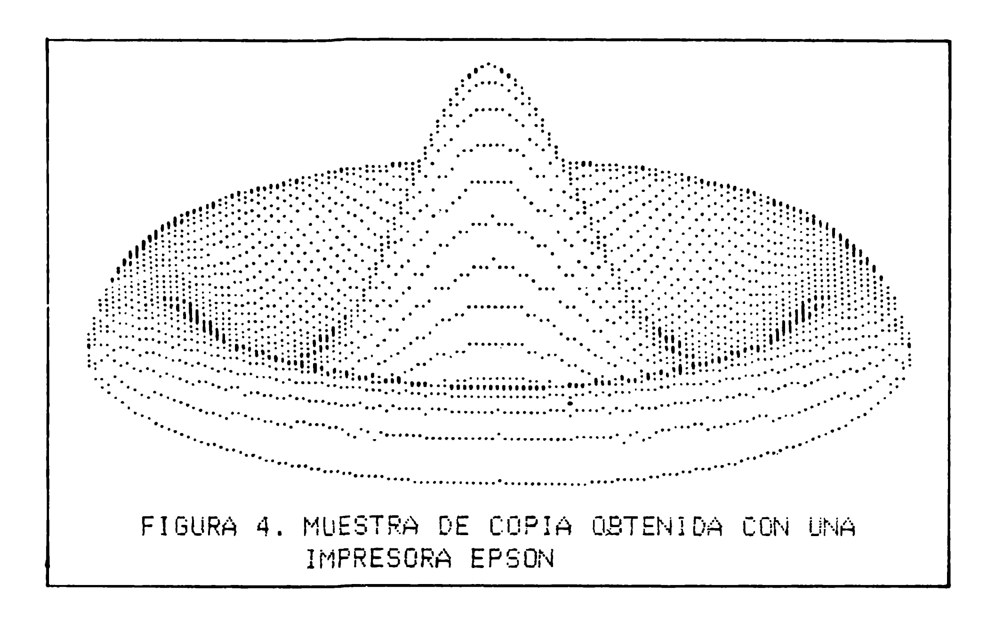


FIGURA 4. MUESTRA DE COPIA OBTENIDA CON UNA IMPRESORA EPSON

Otras utilizaciones del RS232

El canal «b» puede hacerse servir también para transmitir programas, datos, código máquina y pantallas de forma similar a como lo hacen los canales de tipo «m» (aunque el canal «b» no necesita los nombres de los ficheros). Esta posibilidad puede resultar particularmente útil para enviar programas por teléfono si, por ejemplo, se adquiere un modem RS232. Disponiendo del hardware correspondiente, para enviar programas mediante un modem de 300 baudios a un amigo nuestro que esté al otro extremo del hilo telefónico, deberá hacerse:

```
FORMAT "b";300: LOAD*"b"
```

(Esto deberá entrarlo nuestro amigo y deberá ser lo primero que se haga). De esta manera, el Spectrum se prepara a esperar al programa. Entonces nosotros tenemos que escribir:

```
FORMAT "b";300: SAVE*"b"
```

El canal «t» debería dedicarse únicamente a la transmisión de textos o listados. Todos los comandos LOAD*, SAVE*, MERGE* y

VERIFY* pueden emplearse con el RS232 y lo mismo ocurre con OPEN # y CLOSE #. En los canales relacionados con el RS232 no es necesario especificar los nombres de los ficheros ni los números que les siguen. De todas maneras, el Spectrum los acepta aunque los ignora. Por ejemplo, se aceptará SAVE* "b"; 5; "nombre" pero el resto de detalles se ignorará.

Otra aplicación puede ser la que consiste en que el Spectrum actúe como terminal de otro ordenador de forma que los caracteres que se pulsen en el Spectrum sean transmitidos al ordenador.

```
10 FORMAT "b";BAUD
20 OPEN #4;"b"
30 PAUSE 0: LET a$=INKEY$
40 IF a$>CHR$ 127 THEN GO TO 30
50 IF a$=CHR$ 6 THEN POKE 23658,
8-PEEK 23658: GO TO 30
60 PRINT #4;a$;
70 GO TO 30
```

El programa trabaja por exploración del teclado (línea 30). Ignora todas las palabras especiales (por ejemplo STOP) que no sean los estándars (40) y recurre a CAPS LOCK para pasar del modo C al L (línea 50). Si el carácter es válido, se envía a la corriente 4 (el ordenador que recibe los datos). Algunos ordenadores pueden tener reacciones extrañas ante alguno de los códigos del Spectrum (como, por ejemplo, ante INV VIDEO). Por lo tanto, será necesario que se realicen algunas comprobaciones adicionales para el Spectrum.

Otra aplicación podría ser la siguiente en la que el Spectrum actúa como terminal de presentación de datos (display). Se imprime en el Spectrum todos los datos llegados a través del RS232.

```
10 FORMAT "b";BAUD
20 OPEN #4;"b"
30 LET a$=INKEY$ #4
40 IF a$=""THEN GO TO 30
50 IF a$=CHR$ 12 THEN CLS : GO T
O 30
60 PRINT a$;
70 GO TO 30
```

Cuando se emplea el RS232 para entrada de datos la pulsación de la tecla SPACE provocará por sí sola un **Break**: no será necesario pulsar CAPS SHIFT.

Es posible conectar dos Spectrums a través del RS232 y con un cable apropiado. Sin embargo, resulta más ventajoso emplear el enlace para red puesto que, de esta manera, se consigue el mismo efecto más rápidamente al tiempo que el RS232 queda libre para conectar otros periféricos.

Ampliación del «Catálogo de corrientes» teniendo en cuenta al RS232

Si se quiere que el programa «Catálogo de corrientes» sea capaz de gestionar los canales relacionados con el RS232 hay que añadir las líneas siguientes:

«Catálogo de corrientes 3»

```
1640 IF F$="T" OR F$="B" THEN  G
O TO 3000
2999 REM CANAL T/B
3000 PRINT  INVERSE 1;"RS232  ";
3010 IF FN P(D+5)=3130 OR FN P(D
+5)=3132 THEN  PRINT "TEXTO": GO
 TO 3040
3020 IF FN P(D+5)=3335 OR FN P(D
+5)=3162 THEN  PRINT "BINARIO":
GO TO 3040
3030 PRINT "(DESCONOCIDO)"
3040 GO SUB 1800
3050 PRINT "VELOCIDAD EN BAUDIOS
:";INT (3500000/((FN P(23747)+2)
*26));" (aprox)"
3060 GO TO 1500
```

Ambos tipos de canales RS232 tienen el mismo identificador en memoria: «T». Por lo tanto, las líneas 3010 y 3020 se encargan de investigar si se trata de un canal de texto o de un canal binario. El programa imprime también la velocidad en baudios.

Capítulo 7

UTILIZACION DE LA RED DE AREA LOCAL

Las redes permiten interconectar varios ordenadores de manera que se comuniquen entre ellos rápidamente. En el Spectrum la velocidad de comunicación es de más de 3 K bytes por segundo que resulta bastante mayor que la velocidad superior del RS232 (19200 baudios). Se pueden conectar hasta 64 Spectrums. Esta posibilidad ofrece varias posibles aplicaciones. Una de ellas es la interconexión de varios Spectrums con fines educativos en una misma clase escolar. De esta manera se pueden compartir periféricos como impresoras y Microdrives. Otra aplicación es la que consiste en comunicar los Spectrums de dos amigos de manera que puedan intercambiarse rápidamente programas y datos. Finalmente, se puede hacer uso de la red para facilitar el desarrollo de programas (especialmente en código máquina). Uno de los Spectrums puede tener el ensamblador y los programas de desarrollo y los puede transmitir al otro Spectrum. La ventaja de este sistema de trabajo es que si el segundo Spectrum sufre un error «fatal» puede recibir rápidamente una versión modificada del programa desde el primer Spectrum.

Para emplear la red se utilizan canales y corrientes con el identificador "N" o "n" por "Network" (Red en inglés). Antes de emplear la red hay que decidir qué número se asigna a nuestro Spectrum (que dentro de la red es una «estación»). Esto se consigue con la sentencia:

```
FORMAT "n";t
```

donde **t** es el número deseado (entre el 1 y el 64). Al conectar el Spectrum, se le asigna el número 1. Si sólo hay dos Spectrums en

la red no es necesario realizar una instrucción FORMAT en ambos puesto que cualquiera de ellos puede ser la estación 1. Si ha olvidado qué número de estación le corresponde, la función:

```
PEEK 23749
```

se lo dirá.

Para conectar las estaciones hay que emplear los cables que acompañan cada Interface y conectarlos a los zócalos que hay en él —debajo de los zócalos del cassette— formando así una cadena. Hay que dejar siempre un cable libre puesto que se debe evitar la creación de un bucle cerrado (la **figura 5** muestra los detalles de la conexión de tres estaciones).

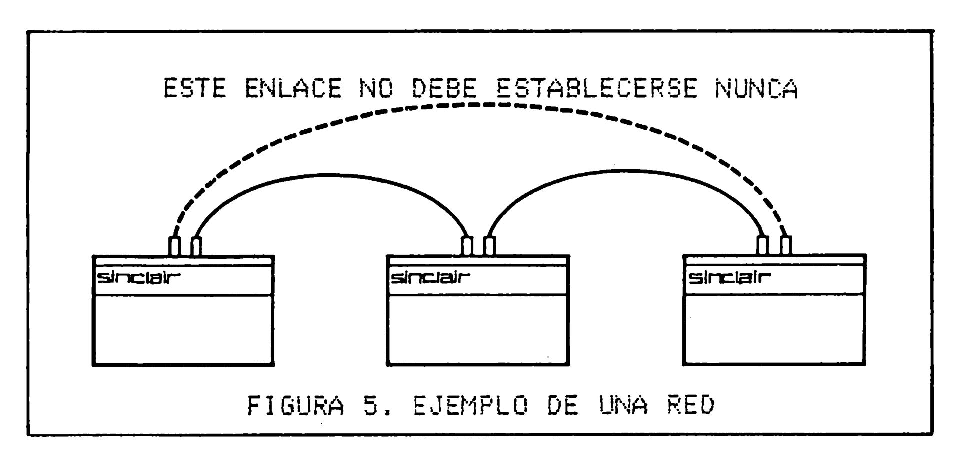


FIGURA 5. EJEMPLO DE UNA RED

Transmisión de programas a través de la red de área local

Al igual que con los Microdrives y el RS232, es posible enviar programas y datos a través de la red. En este caso las instrucciones necesarias requieren la escritura del identificador «N» seguido del número de una estación. Por ejemplo, si su Spectrum es la estación 10 y un amigo suyo está en la 20 habría que indicar:

```
SAVE *"n";20
```

para enviarle un programa. El borde de su pantalla tendría el color

negro mientras espera que su amigo acepte el programa. Su amigo debería indicar:

```
LOAD *"n" ;10
```

para aceptar el programa. Cuando lo haga, los bordes de la pantalla de ambas estaciones empezarán a parpadear y no dejarán de hacerlo hasta que no se haya transferido completamente el programa. Si su amigo hubiera hecho LOAD primero, su Spectrum tendría que haber esperado a que usted hiciera SAVE. Después de esto, el programa se habría transferido perfectamente. Por razones que explicaremos más adelante resulta muy conveniente que el receptor haga LOAD antes de que el emisor escriba SAVE.

Si su amigo quiere confirmar que el programa se ha transferido correctamente, tendrá que hacer:

```
VERIFY *"n" ;10
```

y usted tendrá que introducir:

```
SAVE *"n" ;20
```

De nuevo ambas operaciones podrían haberse realizado en uno u otro orden. Se pueden añadir a estas instrucciones todas las funciones normales que ya se han utilizado anteriormente. Por ejemplo:

```
SAVE *"n" ;10 LINE 10
LOAD *"n" ;20 SCREEN$
```

Transmisión de datos a través de la red de área local

Para enviar datos a una estación se utilizan las corrientes y el canal «N». Para abrir una corriente se hace servir la sentencia:

```
OPEN #s; "n" ;x
```

donde **s** es el número de la corriente (0—15) y **x** es el número de la otra estación, del 0 al 64 (el 0 tiene una función especial que estu-

diaremos más tarde). Con esto se crea un «buffer» similar al que se empleaba con los canales de tipo «M» pero de un tamaño inferior (la mitad: 255 bytes). Cuando se abre una corriente asociada a la red puede ser un fichero de escritura o de lectura pero no de ambos tipos. Determinamos de qué tipo se trata por la primera acción que realizamos con él: si se hace PRINT # se convierte en un fichero de escritura; si se lee de él (con INPUT # o INKEY$ #) se convierte en un fichero de lectura. Al igual que ocurría con los Microdrives, los comandos CLOSE # son importantísimos con los ficheros de escritura (si no se hace CLOSE los últimos datos del buffer no serán enviados nunca y la estación receptora quedará eternamente a la espera).

Mientras se llevan a cabo transmisiones a través de la red (es decir, mientras parpadea el borde de la pantalla) la tecla CAPS no es necesaria para realizar un **Break** (basta con pulsar SPACE). Nunca hay que conectar o desconectar un Spectrum mientras está teniendo lugar una comunicación.

La estación 0: difusión

Normalmente cuando se envían o reciben datos a/o desde otra estación, se sigue un protocolo de comunicación: si la estación receptora no está preparada, la emisora tendrá que esperar y los datos no se perderán. Sin embargo, se ha dotado al Interface de una característica muy potente que es la de la «**difusión**» empleando el número 0 (que no sigue los protocolos antedichos).

Cuando se envían datos a la «estación» cero se están enviando, en realidad, a todos los Spectrums conectados a la red y no se espera a que indiquen estar dispuestos a recibir la información que se les envía. Si los ordenadores están a la espera de recibir datos de la estación 0, los aceptan pero, en caso contrario, la información se «pierde». Cuando se transmiten programas a través de la red a varias estaciones, cada una de ellas debe haber hecho:

```
LOAD *"n";0
```

y los bordes de sus pantallas serán de color negro mientras espe-

en el programa. Después de esto, la estación que envía el programa debe hacer:

```
SAVE *"n" ;0
```

y todas las estaciones que están «a la escucha» podrán cargar el programa. Cuando se utiliza esta operación de «difusión» desde la estación 0 no hay ningún método sencillo de saber si ha habido alguien que haya recibido la información emitida y, en este caso, tampoco es fácil saber quién ha sido el receptor. Sin embargo, cuando alguien lee cualquier información procedente de la red, puede saber quién se la ha enviado haciendo:

```
PEEK 23759
```

con la que se obtendrá el número de la estación emisora.

El programa de servicio de la impresora y el Microdrive

Una de las principales ventajas de la red es que se pueden compartir periféricos —como por ejemplo, impresoras y Microdrives— entre varios Spectrums. Aquí van dos programas que permiten a todas las estaciones secundarias (en una clase, las de los alumnos) hacer uso de la impresora y los Microdrives de la estación principal (en el mismo ejemplo, la del profesor). Los programas se han escrito con el objetivo de que la utilización de ambos periféricos resulte lo más fácil posible mediante el empleo de comandos como SAVE y la LPRINT. Uno de ellos es para la estación principal y el otro para cada una de las secundarias.* Para empezar, cada estación secundaria tiene que cargar el programa que le corresponde. La forma más sencilla de conseguirlo es emitiendo el programa desde la estación principal a cada una de las secundarias. De esta forma será recibido por todas las demás estaciones. Una vez hecho esto, hay que cargar el programa principal en la estación que disponga de la impresora y de los Microdrives. Luego se ejecutará el programa

---

* Nota del T.: Denominamos estación *principal* a la que en ocasiones se le llama *maestra.* Damos el nombre de *estación secundaria* a la que también se llama *esclava.*

mediante RUN. Una vez se haya cargado todo el código máquina gracias a las diversas instrucciones POKE, el borde de la pantalla de la estación tomará el color negro y el sistema estará listo para ser utilizado.

Cada estación secundaria puede recurrir a la instrucción FORMAT para asignarse un número de estación (excluidos el 32 y el 64). Mientras los números de línea se mantengan por debajo de la cifra 9000 (para evitar solapamientos con los programas secundarios) los programas pueden escribirse y corregirse en las propias estaciones secundarias. Si una de ellas desea emplear uno de los periféricos de la estación principal tendrá que hacer **GOTO 9000.** Inmediatamente aparecerá un menú y el usuario podrá hacer servir la impresora, catalogar, cargar o almacenar sus programas.

Si el usuario elige la opción 1, quedará conectado a través de la estación principal a cualquier impresora —ZX o del tipo RS232— unida a ella. A partir de ese momento podrá utilizar LPRINT, LLIST, etc., hasta el momento en que termine (instante en el que deberá escribir CLOSE # 3). La opción 2 permite almacenar (SAVE) cualquier programa o fichero de cualquier tipo (por ejemplo, SCREEN$) en un cartucho de la estación principal. Con la opción 3 se puede cargar (LOAD) cualquier fichero situado en el cartucho de la estación principal y que previamente haya sido almacenado con la opción 2. La opción 4 imprime un catálogo de todos los programas del usuario. La opción 5 se emplea para indicar que se desea terminar de trabajar.

Listado del programa secundario

```
9000 CLS
9010 PRINT '"  1. UTILIZACION IM
PRESORA"
9020 PRINT '"  2. ALMACENAR UN P
ROGRAMA           (SAVE)"
9030 PRINT '"  3. CARGAR UN PROG
RAMA (LOAD)"
9040 PRINT '"  4. CATALOGAR EL C
ARTUCHO (CAT)"
9050 PRINT '"  5. FINALIZAR"
9060 PAUSE 0: LET a$=INKEY$
```

```
9070 IF a$="1" THEN  GO TO 9150
9080 IF a$="2" THEN  GO TO 9300
9090 IF a$="3" THEN  GO TO 9400
9100 IF a$="4" THEN  GO TO 9500
9110 IF a$<>"5" THEN  GO TO 9060
9120 GO TO 9000
9147 REM ************************
9148 REM *ENVIAR A LA IMPRESORA*
9149 REM ************************
9150 LET a$="IMPRIMIR": GO SUB 9
200
9160 CLOSE #3: OPEN #3;"n";64
9170 PRINT "LA IMPRESORA ESTA LI
STA. USAR  LPRINT, LLIST ETC. A
L FINALIZAR HACER CLOSE #3"
9180 GO TO 16000
9197 REM ************************
9198 REM * EMITIR A$ Y ESPERAR *
9199 REM ************************
9200 PRINT  INVERSE 1;"SE ESTABL
ECE CONTACTO CON LA    ESTACION
PRINCIPAL."
9205 CLOSE #4: OPEN #4;"n";0
9210 PRINT #4;a$
9220 CLOSE #4
9230 OPEN #4;"n";64
9240 IF INKEY$#4="" THEN  GO TO
9205
9250 CLOSE #4
9260 PRINT "CONTACTO CONSEGUIDO.
"
9270 RETURN
9297 REM ************************
9298 REM *ALMACENAR UN PROGRAMA*
         *      (SAVE)         *
9299 REM ************************
9300 LET a$="save": GO SUB 9200
9310 INPUT "NOMBRE DEL FICHERO A
 ALMACENAR:"; LINE f$
9320 OPEN #4;"n";64
9330 PRINT #4;f$
```

```
9340 CLOSE #4
9350 PRINT "MICRODRIVE LISTO. HA
CER LOAD *""n"";64 ETC"
9360 GO TO 16000
9397 REM **********************
9398 REM * CARGAR UN PROGRAMA *
         *       (LOAD)       *
9399 REM **********************
9400 LET a$="load": GO SUB 9200
9410 INPUT "NOMBRE DEL FICHERO A
 CARGAR:"; LINE f$
9420 OPEN #4;"n";64
9430 PRINT #4;f$
9440 CLOSE #4
9450 OPEN #4;"n";64
9460 INPUT #4;st
9470 CLOSE #4
9480 IF st<>6 THEN  PRINT "NO SE
 HA ENCONTRADO EL FICHERO": STOP

9490 PRINT "MICRODRIVE LISTO. HA
CER LOAD *""n"";64 ETC"
9495 GO TO 16000
9497 REM ***********
9498 REM * CATALOGO *
         *   (CAT)  *
9499 REM ***********
9500 LET a$="cat": GO SUB 9200
9510 CLS
9520 PRINT "CATALOGO:"
9530 MOVE "n";64 TO #2
9540 PRINT ' FLASH 1;"PULSAR CUA
LQUIER TECLA PARA     SEGUIR"
9550 PAUSE 0
9560 GO TO 9000
```

Listado del programa principal

```
1 REM ***********************
2 REM PROGRAMA DE LA ESTACION
              PRINCIPAL
3 REM ***********************
```

```
  10 CLEAR 65089
  20 LET st=65260: GO SUB 8000
  30 LET open=65090: GO SUB 8350
  40 DEF FN p(p)=PEEK p+256*PEEK
 (p+1)
 100 FORMAT "n";64
 110 CLOSE #4
 120 OPEN #4;"n";0
 130 INPUT #4; LINE a$
 140 CLOSE #4
 150 LET n=PEEK 23759
 160 IF a$="print" OR a$="save"
OR a$="load" OR a$="cat" THEN  G
O TO 180
 170 GO TO 120
 180 OPEN #4;"n";n
 190 PRINT #4
 200 CLOSE #4
 210 IF a$="save" THEN  GO TO 30
0
 220 IF a$="load" THEN  GO TO 60
0
 230 IF a$="cat" THEN  GO TO 800
 237 REM ***********************
 238 REM *PETICION DE IMPRESORA*
 239 REM ***********************
 240 PRINT "LA IMPRESORA ESTA SI
ENDO UTILIZADA POR LA ESTACION "
;N
 250 MOVE "n";n TO #3
 260 LPRINT ''''''''
 270 PRINT "(FIN DE IMPRESION)"
 280 GO TO 110
 297 REM *****************
 298 REM *PETICION DE SAVE*
 299 REM *****************
 300 OPEN #4;"n";n
 310 PRINT "LA ESTACION ";n;" ES
TA ALMACENANDO"
 320 DIM f$(10): INPUT #4; LINE
f$
```

```
 330 CLOSE #4
 340 GO SUB 500: LET f$(10)=CHR$
 n
 350 FOR i=1 TO 10
 360 POKE USR "a"+i-1,CODE f$(i)
 370 NEXT i
 380 POKE 23766,d: POKE 23768,5
 390 LET a=USR open: CLOSE #5
 400 IF a<>1 THEN  ERASE "m";d;f
$
 410 OPEN #5;"m";d;f$
 420 POKE FN p(5*2+23566+8)+FN p
(23631)-1+67,4
 430 MOVE "n";n TO #5
 440 CLOSE #5
 450 PRINT "(PROGRAMA ALMACENADO
)"
 460 GO TO 110
 497 REM ***********************
 498 REM *     CALCULAR d A     *
         *     PARTIR DE n      *
 499 REM ***********************
 500 LET d=1
 510 RETURN
 597 REM ***********************
 598 REM *CARGA DE UN PROGRAMA*
         *        (LOAD)        *
 599 REM ***********************
 600 OPEN #4;"n";n
 610 DIM f$(10): INPUT #4; LINE
f$
 620 CLOSE #4: LET f$(10)=CHR$ n
 630 PRINT "LA ESTACION ";n;" ES
TA CARGANDO"
 640 GO SUB 500
 670 FOR i=1 TO 10
 680 POKE USR "a"+i-1,CODE f$(i)
 690 NEXT i
 700 POKE 23766,d: POKE 23768,5
 710 LET a=USR open
 720 OPEN #4;"n";n
```

```
 730 PRINT #4;a
 740 CLOSE #4
 750 IF a<>6 THEN  GO TO 770
 760 MOVE #5 TO "n";n
 770 CLOSE #5
 780 PRINT "(FIN DE CARGA DEL PR
OGRAMA)"
 790 GO TO 110
 797 REM *********************
 798 REM *   CATALOGO DE UN   *
         *   CARTUCHO (CAT)   *
 799 REM *********************
 800 PRINT "CATALOGO (CAT) PARA
LA ESTACION ";n
 805 GO SUB 500
 810 LET z$="": CAT #14,d
 820 OPEN #4;"n";n
 830 LET z$=z$(13 TO )
 840 IF LEN z$<10 THEN  GO TO 88
0
 850 IF z$(10)=CHR$ n THEN  PRIN
T #4;z$( TO 9)
 860 LET z$=z$(12 TO )
 870 GO TO 840
 880 PRINT #4: CLOSE #4
 890 PRINT "(FIN DE OPERACION CA
T)"
 900 GO TO 110
 910 REM
```

Obsérvese que el programa principal necesita que las rutinas "OPEN # cualquier cosa" y "Corriente 14-z$" estén presentes a partir de las líneas 7995-8510.

Explicaré los dos programas al mismo tiempo para que usted pueda saber qué ocurre en ambas estaciones.

Lo primero que hace la estación principal es activar los programas en código máquina. Con estaciones principales de 16 K se deberá realizar la sustitución:

```
10 CLEAR 32489
20 LET st=32490: GO SUB 8000
30 LET open=32490: GO SUB 8350
```

La estación principal debe tener el número 64. Esto queda fijado en la línea 100.

La estación secundaria debe, antes que nada, ponerse en contacto con la principal. Para ello, difunde una palabra a través de la red. La estación principal, una vez la ha recibido, la reconoce. Los ficheros de las estaciones secundarias se almacenan en cartucho empleando las nueve primeras letras para el nombre. La décima es CHR$ N donde N es el número de la estación secundaria. De esta manera, se pueden mantener separados uno de otro los programas de las estaciones secundarias.

Cuando una estación secundaria hace uso de la opción 1 para conseguir el acceso a la impresora, emite repetidamente la palabra «print» hasta recibir un carácter de la estación 64. El establecimiento de este contacto queda encargado a la rutina de la línea 9200 que abre la corriente 4 como corriente de difusión (9205), envía **a$** (9210) y luego cierra la corriente. Después comprueba que haya habido respuesta de la estación 64 (9230-9240). Si no la ha habido se vuelve a difundir **a$**. Una vez recibida contestación, la línea 9610 procede a abrir la corriente 3 para que envíe datos a la estación principal. El **GO TO 16000** que hay en la línea 9180 y en otras tiene la misión de parar el programa y dar el mensaje **Program finished** (Programa terminado).

El programa principal está leyendo continuamente cadenas de la corriente de difusión (líneas 120-140). Una vez leída cada una de ellas comprueba si se trata de una petición de alguna estación secundaria (160). En caso de que no sea así, continúa leyendo nuevas cadenas. Si efectivamente se trata de una petición procedente de una estación secundaria, el programa pasa a calcular el número de la estación (150) y le envía una respuesta (180-200). Entonces se pasa a descubrir qué palabra de petición, entre las varias posibles, se ha recibido y, una vez conocida, se ejecuta el GOTO correspondiente (210-230). Si la palabra difundida era «print», se imprime el mensaje adecuado en el televisor de la estación principal y se emplea la potente instruccion MOVE para transferir todas las entradas procedentes de la estación secundaria al dispositivo que en ese momento esté conectado a la corriente 3 (línea 250). Cuando una estación secundaria envía la instrucción CLOSE # 3, termina el efecto de la instrucción MOVE y se imprime en la pantalla un nuevo mensaje.

A partir de este momento, el proceso empieza de nuevo y la estación principal centra su atención otra vez en el canal de difusión.

Cuando una estación secundaria elige la opción 2 para almacenar (SAVE) un programa, difunde la palabra «save» hasta que la estación principal responda (9300). Se le pide entonces el nombre del fichero. Una vez dado, se envía al maestro (9310-9340). Se imprime un mensaje y se puede hacer un SAVE*"n";64 seguido del tipo del fichero. Si sólo se van a almacenar programas hay que realizar la sustitución:

```
9350 SAVE *"n";64: GO TO 9000
```

En cuanto la principal recibe la palabra «save» de una estación secundaria a través del canal de difusión abre una corriente que vaya a la estación secundaria e imprime un mensaje (300-310). La línea 340 emplea la subrutina de la línea 500 para calcular cuál es el Microdrive que corresponde a la estación que realiza la llamada. En la versión que se ilustra en estas páginas siempre se selecciona la unidad 1. Si el número de estaciones secundarias es grande, sin embargo, este método de selección puede no resultar práctico. Si, por ejemplo, hay 63 estaciones secundarias y la principal dispone de ocho unidades de cartucho, cada una de las unidades podría compartirse entre ocho estaciones secundarias. En ese caso, la línea 500 sería de la forma:

```
500 LET d=1+INT (n/8)
```

Una vez se ha calculado el número de la unidad de cartucho se le asigna al décimo carácter del nombre del fichero el número de la estación a la que pertenece (línea 340). Con un POKE se envía ese carácter a una posición de la memoria contenida en el área de gráficos definidos por el usuario (350-370). Esto, conjuntamente con la línea 380, lleva a cabo los preparativos necesarios para abrir cualquier rutina (OPEN #) a la corriente 5 (390). Si el fichero al que se hace referencia ya existía, se borra 400. La línea 410 abre un fichero de escritura con el nombre adecuado y la línea 420 se encarga de engañar al sistema haciéndole creer que no se trata de un fichero de datos. Utilizando nuevamente MOVE se escribe en el

fichero el programa enviado desde la estación secundaria (430). Cuando se ha terminado el SAVE se cierra la corriente y se imprime un mensaje (440-450).

En caso de la estación secundaria elija la opción 3 (cargar un programa) se envía la palabra «load» y se espera una contestación (9400). Una vez recibida, se pide el nombre de un fichero, se envía a la estación maestra y se comprueba si ese fichero existe o no. En caso negativo, se genera el error correspondiente (9480). Si el fichero existe se puede cargar mediante:

```
LOAD *"n";64
```

Si sólo se van a cargar programas, hay que realizar la sustitución:

```
9490 LOAD *"n";64: GO TO 9000
```

Cuando la estación principal recibe la palabra «load» (220) lee el nombre del fichero que le pide la estación secundaria (600-620) y hace que el décimo carácter indique su procedencia. Se calcula el número de la unidad de cartucho (640) y se transfiere mediante POKEs el nombre del fichero al área de gráficos definidos por el usuario. La línea 700 realiza unos cuantos POKEs más antes de pasar a utilizar la rutina "OPEN # cualquier cosa" (710) para comprobar la existencia del programa indicado. El número que se obtiene se devuelve a la estación secundaria. Sólo en el caso de que el tipo del fichero sea el correcto se cargará el fichero para transferirlo del cartucho a la estación secundaria mediante la instrucción MOVE (760). Luego la corriente del Microdrive se cierra a la vez que se imprime un mensaje (770-780).

Si una estación secundaria escoge la opción 4 para catalogar sus programas, emitirá la palabra «cat» hasta que se reciba la confirmación de que ha sido aceptada (9500). En ese instante se borra la pantalla, se imprime en ella un título y luego aparece toda la entrada procedente de la estación maestra (esto es, los nombres de los ficheros) recurriéndose nuevamente a la instrucción MOVE (9530).

Cuando la estación principal recibe una petición «cat» (230), imprime un mensaje y calcula el número de la unidad de cartucho (800-805). Gracias a la rutina «Corriente 14-z$» se envía un CATá-

logo a z$ (810) y se abre una corriente hacia la estación secundaria (820). Se quita el título de la cadena (830) y, si lo que sigue es el nombre de un cartucho, se inspecciona el valor del décimo carácter. Si resulta ser CHR$ N —es decir, el programa de la estación secundaria— entonces se envía el nombre a la estación secundaria (850). Cuando ya no quedan más nombres de ficheros, se cierra la corriente y se imprime un mensaje (880-890).

Nuevas mejoras

Algunos usuarios pueden necesitar incluir nuevas posibilidades en estos programas. Una mejora podría ser la de asociar a cada estación una lista de contraseñas (almacenadas en un conjunto o en un fichero de datos) de forma que ninguna estación pudiera hacerse pasar por otra y, destruir los programas de otro usuario. Si, además, se guardaran los nombres de los usuarios de cada estación, los programas al salir por la impresora podrían llevarlos como encabezamiento, haciéndose más fácil la identificación de listados.

Ampliaciones de «Catálogo de corrientes» para la red

Para que el programa «catálogo de corrientes» sea capaz de reconocer las corrientes de la red, hay que añadir las siguientes líneas de programa. Con ellas será posible imprimir el número de la estación emisora y el de la receptora. Asimismo, se indicará cuándo se está empleando la posibilidad de emisión y cuándo no.

```
1650 IF F$="N" THEN  GO TO 3500
3499 REM CANAL N
3500 PRINT  INVERSE 1;"RED"
3510 GO SUB 1800
3520 PRINT "LA ESTACION NUM. :";
PEEK 23749
3530 PRINT "ESTA TRANSMITIENDO A
 :";PEEK (D+11);" (EMITIENDO)" A
ND PEEK (D+11)=0
3540 GO TO 1500
```

Capítulo 8

EL INTERFACE Y EL LENGUAJE MAQUINA

Este Capítulo está dirigido a aquellos lectores que ya posean algún conocimiento acerca del lenguaje máquina del microprocesador Z80. Los lectores que no lo conozcan podrán, sin embargo, sacar provecho de algunas secciones.

La ROM adicional de 8 K

Además de los 16 K de ROM del Spectrum, el Interface tiene una ROM de 8 K que está paginada sobre la misma área de memoria, es decir, la que va del 0000 al 1FFFH (los números seguidos de una H están en hexadecimal). El acertado diseño del mecanismo de paginación hace que al sistema operativo del Spectrum le resulte muy fácil emplear las rutinas adicionales guardadas en la ROM del Interface. Sin embargo, el mismo mecanismo complica bastante la vida a los programadores en código máquina que quieran utilizarlas. Antes de entrar más a fondo en el sistema operativo y la ROM hay que explicar el inteligente sistema de paginación que se utiliza. Para evitar confusiones todas las posiciones de la memoria ROM adicional van precedidas en este libro por una «X». En el caso de que las direcciones sean distintas para cada ROM se escribe primero la posición antigua y luego la nueva, separadas ambas por una barra inclinada.

El mecanismo de paginación

Con una sola excepción, la paginación de la ROM se produce cuando el contador del Z80 alcanza la dirección 0008, es decir, cuan-

do se produce un RST 8. En la ROM del Spectrum —la de 16 K— esta instrucción (restart) de reanudación va seguida de un byte de datos que determine el mensaje de error que se ha de comunicar durante la ejecución de un programa en BASIC (o bien indica que ha de aparecer un signo de interrogación intermitente para indicar que la sintaxis de una línea no es correcta). Si se intenta emplear cualquiera de las instrucciones especiales (por ejemplo CAT) o de las instrucciones de sintaxis extendida (por ejemplo LOAD*) el BASIC de 16 K provocará un error utilizando RST 8. Si el Interface está conectado esto hará que se saque el BASIC y se pagine la ROM del Interface. Entonces se calcula qué fue lo que causó el error y, en particular, si lo provocó una operación extendida del BASIC. Si no se trata de este último caso se llama a la rutina normal de tratamiento de errores (que está en la ROM de 16 K). Si la causa del error fuera una instrucción extendida, la ROM adicional pasaría a actuar (ya sea ejecutando la instrucción ya sea comprobando su sintaxis). Cuando terminara de tratar el error, la ROM dejaría de ocupar el espacio de memoria sobre el que habría sido paginada. Hay que decir que éste es un método ciertamente brillante para añadir nuevas instrucciones sin tener que tocar un solo byte de la ROM de 16 K del Spectrum. Tiene como desventaja la de dificultar el uso de las rutinas de la ROM adicional en los programas que uno desarrolla.

Para superar este obstáculo, los diseñadores de la ROM adicional han empleado lo que llamaremos "códigos especiales". Estos códigos permiten a las rutinas tener acceso a determinadas rutinas de la ROM adicional. Estos códigos especiales son bytes de datos que van detrás de la instrucción RST 8 y toman los valores que van del 1BH al 32H, ambos inclusive. Los códigos que van del FFH al 1AH producen los mensajes de error normales, mientras que los códigos superiores al 32H causan un error del tipo **Hook/ code error** que no están recogidos en el manual del Spectrum.

Algunos de estos códigos son muy útiles mientras que otros no lo son tanto. El programador ha de tener muy en cuenta el estado de las interrupciones enmascarables («Maskable Interrupts») especialmente al entrar o salir de algunas subrutinas. Si se ha de realizar un retorno al BASIC (después de una función USR) entonces el valor del par de registros H´L´ tiene que conservarse (siempre que

se utilicen rutinas de los Microdrives). Si no se conservaran, podría quedar afectado de forma negativa el calculador en coma flotante, lo que podría repercutir gravemente en el sistema.

Quizás el código más útil entre los especiales sea el 32H aunque oficialmente se le designe como «código reservado para futura expansión por SRL». Así se oculta el hecho de que este código (Sinclair Research Laboratories) permite que una rutina del usuario pueda llamar a la mayor parte de las rutinas de la ROM adicional al mismo tiempo que permite controlar la paginación de ésta a voluntad.

Algunos lectores pueden estar interesados en desensamblar la ROM de 8 K. Haciendo un PEEK o un LD A, (HL) sólo se podrá leer la ROM de 16 K. Para desensamblar completamente la ROM de 8 K hay que hacer lo que yo programé con un Spectrum de 48 K:

```
10 CLEAR 32767
20 SAVE *"m";1;"8K ROM"CODE 0,81
92
30 LOAD *"m";1;"8K ROM"CODE 32768
```

Cuando se ejecuta el SAVE* la ROM adicional está en su lugar de forma que conserva una copia suya en el cartucho. La línea 30 simplemente vuelve a cargar el contenido de la ROM a partir de la posición 8000H (por encima de la posición que normalmente ocuparía). Entonces es posible desensamblar por completo la ROM a partir de la posición 8000H teniendo siempre en cuenta que hay que restar 8000H a todas las direcciones absolutas.

Aparte de cuando se realiza un RST 8, la paginación de la ROM adicional tiene lugar cuando el PC es igual a 1708H (posición que corresponde a la mitad de la rutina asociada a la instrucción CLOSE #). Esto se debe a que contiene un error «fatal» y a que resulta muy poco adecuada para trabajar con los tipos adicionales de canales que utiliza el Interface.

La ROM adicional anterior y la nueva

En marzo de 1984 Sinclair rediseñó la ROM adicional que formaba parte de las unidades de Interface 1. Aunque las diferencias son

prácticamente transparentes para quien programe en BASIC, el programador en lenguaje máquina las advertirá. En la nueva ROM se han corregido la mayoría de errores de la ROM anterior, en especial los que tenían que ver con CAT y FORMAT. Sin embargo, la mayor parte de las rutinas están situadas en lugares distintos de cada ROM con lo que su utilización difiere ligeramente. Para saber qué tipo de ROM está conectada mientras está paginada (ver más adelante para saber cómo se hace esto) hay que comprobar la posición X16DA. En la ROM anterior esta posición se encuentra en una sección no utilizada y contiene el valor FFH pero en la nueva memoria es el inicio de un mensaje de «copyright» y su contenido es 7FH. Siempre que ha sido posible se ha utilizado un código independiente de la posición, para la confección de las notas siguientes de la ROM.

Cómo funcionan los canales y las corrientes

Los canales y las corrientes pueden tratarse desde programas en código máquina de forma similar a como se hacía desde el BASIC. Las corrientes emplean el área de memoria denominada STRMS y están en contacto con los canales (que emplean el área de memoria CHANS). El área STRMS es fija y está comprendida entre las posiciones 5C10H y 5C35H (ambas inclusive). La longitud de esta área es de 19 pares de bytes, uno por cada una de las corrientes FHD a 0FH. Con cada par de bytes se obtiene un desplazamiento de 16 bits. Este desplazamiento tiene el valor cero si no hay ningún canal asociado a la corriente. En caso de que el desplazamiento obtenido no sea cero, se utilizará para realizar un direccionamiento indexado sobre el área CHANS. Cuando se inicializa, el área CHANS está formada por veinte bytes organizados en cuatro grupos de cinco bytes (cinco bytes por canal). La variable del sistema «CHANS» apunta al principio del área de memoria CHANS. Esta posición inicial es la 54FH. El final del área CHANS se encuentra dos bytes antes de PROG.

Inicialmente la información referente a los canales agrupa datos acerca de cuatro canales —cinco bytes para cada canal— según el orden «K», «S» «R» y «P». Los primeros 16 bytes del área STRMS

contienen la información que se muestra en la siguiente tabla junto a los canales correspondientes:

NUMERO DE CORRIENTE	POSICION DE CORRIENTE	DESPLAZAMIENTO DEL CANAL	CANAL
FD	5C10	0001	«K»
FE	5C12	0006	«S»
FF	5C14	000B	«R»
00	5C16	0001	«K»
01	5C18	0001	«K»
02	5C1A	0006	«S»
03	5C1C	0010	«P»
04	5C1E	0000	NINGUNO

Para encontrar los datos del canal correspondiente a una determinada corriente hay que sumar a CHANS (después de comprobar que no es cero) el desplazamiento correspondiente al canal y, luego, restar uno a lo que se obtenga. El resultado apunta a la posición donde empiezan los datos que corresponden. Éstos ocupan, al menos, cinco bytes. El primer par de bytes apuntan a la rutina de salida de la ROM. Los dos segundos apuntan a la rutina de entrada y el quinto byte es el identificador —en mayúsculas—del canal. Por ejemplo, el canal «S» se almacena de la siguiente manera:

```
CHANS+5    DEFW 09F4H
           DEFW 15C4H
           DEFM "S"
```

En realidad, 09F4H se utiliza como rutina de salida de los canales «K», «S» y «P» aunque las acciones que se realizan son distintas según cual sea el identificador del canal.

Los canales del Interface

Para hacer servir cualquiera de las posibilidades del Interface el sistema necesita traer de la posición 5CB6H 58 bytes adicionales

para sus variables. Hay que inicializar esta área antes de usar cualquier rutina de la ROM adicional o de los códigos especiales y antes de que se creen tipos de canales adicionales (como por ejemplo, la rutina «Corriente 14-z$»). Esto se consigue mediante las instrucciones:

```
CF RST 08
31 DEFB 31H
```

Si el Interface aún no está conectado, se producirá el error «i». En caso contrario, se insertarán los 58 bytes y no habrá error alguno. Las variables utilizadas se detallan en el Apéndice A.

Las variables adicionales del sistema se crean generalmente cuando:

i) hay un error (ya sea de sintaxis, ya sea de ejecución)
ii) se introduce o ejecuta cualquier instrucción del BASIC extendido
iii) se ejecuta una instrucción CLOSE #

Inicialmente, la variable VECTOR de la posición 5CB7H tiene el valor 01F0H pero puede cambiarse para permitir la inclusión de nuevas instrucciones en BASIC. Sólo hablando de este aspecto del Interface se podría escribir un libro entero. Aquí daremos más detalles en un Capítulo posterior.

La variable IOBORD de la posición 5CC6H se utiliza para determinar el color del borde de la pantalla durante determinadas operaciones de Entrada/Salida, es decir las que se realizan a través del RS232, de la red y cuando se escribe en un cartucho. Si usted tiene preferencia por un determinado color puede hacer POKE 23750 con el color de su gusto. Por otra parte, si le molesta el parpadeo del borde puede hacer:

```
POKE 23750,INT (PEEK 23624/8)
```

con el que se consigue que el color del borde durante una operación de E/S sea el que normalmente tiene esta área de la pantalla.

Los canales adicionales del Interface —«M», «N», «T» y «B»— utilizan un formato extendido para almacenar sus detalles en el área CHANS.

El canal «M»

Ocupa 595 bytes del área CHANS. Lo direcciona el registro IX de la ROM adicional. El papel de cada byte es el siguiente (todos los desplazamientos están en decimal):

0	DEFW 8	rutina de salida
2	DEFW 8	rutina de entrada
4	DEFM "M"	identificador del canal
5	DEFW 11D8H/12B3H	salida de la ROM adicional
7	DEFW 1122H/11FDH	entrada de la ROM adicional
9	DEFW 595	longitud del área CHANS
11	CHBYTE	posición dentro del buffer en un momento dado (0-512). (El contador de bytes en curso indica el siguiente byte a añadir o quitar de la zona de datos de 0 a 512 inclusive).
13	CHREC	posición de un registro dentro de un fichero (0-255)
14	CHNAME	los 10 bytes del nombre del fichero
24	CHFLAG	byte indicador. Si el bit 0 de este byte está activado indica que el fichero es de lectura; si está a «1», el fichero es de escritura. Los bits 1 al 7 no se utilizan.
25	CHDRIVE	número de unidad de cartucho (1-8).
26	CHMAP	posición del mapa (MAP) de la unidad Microdrive.
28	HPREAM	12 bytes de preámbulo de encabezamiento. Marca el comienzo del espacio de trabajo del encabezamiento.
40	HDFLAG	byte indicador. bit 0: indica el encabezamiento. bits 1-7: no utilizados (no definidos).
41	HDNUMB	número de sector (0-255)
42		no utilizado (no definido)
44	HDNAME	10 bytes para el nombre del cartucho y espacios a la derecha.
54	HDCHK	Suma de control del encabezamiento.

55 DPREAM	12 bytes previos al bloque de datos (marcan el comienzo del espacio de trabajo de datos).
67 RECFLG	byte indicador. bit 0: no activado para indicar que «no se trata de un encabezamiento». bit 1: no activado: no se ha producido EOF activado: EOF bit 2: no activado: fichero PRINT activado: no se trata de un fichero PRINT. bits 3-7: no utilizados.
68 RECNUM	Número de este registro (0-255)
69 RECLEN	Número de bytes de datos que hay en este registro (0-512)
71 RECNAM	Los diez bytes del nombre del fichero, con espacios a la derecha.
81 DESCHK	Suma de control de la información comprendida entre RECFLG y DESCHK-1
82 CHDATA	El primer byte de un «buffer» de 512 bytes de datos.
593	último byte
594 DCHK	Suma de control de los 512 bytes/precedentes.

Cualquier Microdrive que esté utilizándose tiene, además, 32 bytes para su uso particular, en el área de mapeo de los Microdrives (que va de las posiciones 5CF0H a CHANS-1). Cada bloque de 32 bytes es un mapeo de bits de cada sector del cartucho. Si un determinado bit está activado, significa que el sector no es utilizable bien porque ya contenga datos bien porque esté en malas condiciones o no los haya.

El canal «N»

Ocupa 276 bytes del área CHANS. Lo direcciona el registro IX de la ROM adicional. El papel de cada byte es el siguiente:

0 DEFW 8	
2 DEFW 8	
4 DEFM "N"	identificador del canal
5 DEFW 0D6CH/0EA9H	rutina de salida de la ROM adicional
7 DEFW 0D0CH/0DA9H	rutina de entrada de la ROM adicional.
9 DEFW 276	longitud de esta área CHANS
11 NCIRIS	número de la estación de destino, es decir, a la que se envía el mensaje (0-64)
12 NCSELF	número de la estación de este Spectrum (0-64)
13 NCNUMB	número de bloque (0-65535)
15 NCTYPE	El código o tipo del paquete: 0-datos, 1-EOF (fin de registro)
16 NCOBL	número de bytes que hay en el bloque de datos (0 si se trata de una salida)
17 NCDCS	la suma de control de datos
18 NCHCS	la suma de control del encabezamiento
19 NCCUR	la posición del último carácter tomado de la memoria intermedia («buffer»).
20 NCIBL	el número de bytes en el «buffer» de entrada (0 si se trata de una salida)
21 NCB	inicio del «buffer» de datos de 255 bytes

Los canales «T» y «B»

Estos dos canales ocupan 11 bytes cada uno en el área CHANS (la cantidad mínima para aquellos canales que no sean los normales). El papel de cada byte es:

0 DEFB 8	
2 DEFB 8	
4 DEFB "T"	identificador del canal («T» para la ROM anterior y «B» para la nueva).
5 DEFW 0C3CH/0CEAH(T) o 0C5AH/0D07H(B)	salida de la ROM adicional
7 DEFW 0B6FH/0B76H(T) o 0B75H/0B7CH(B)	entrada de la ROM adicional
9 DEFW 11	longitud de esta área CHANS

Cuando desde el BASIC o desde programas en código máquina se emplean las instrucciones SAVE*, etc. y FORMAT, MOVE o ERASE, se crean canales temporales. Son similares a los canales descritos anteriormente si bien hay cuatro diferencias importantes:

i) el byte del identificador tiene el bit superior activado
ii) estos canales no se conectan nunca a una corriente
iii) desaparecen cuando ha terminado la ejecución de la instrucción que los ha creado o cuando hay algún error.
iv) siempre están al final del área CHANS: un canal permanente nunca puede estar en la memoria más arriba de la posición ocupada por uno temporal.

Los canales que no son los normales tienen un número mínimo de 11 bytes en el área CHANS (tal y como pasa con los canales «T» y «B»). Los dos primeros pares de bytes —normalmente, los punteros a las rutinas de entrada y salida— son ambos iguales a 0008 y ponen en marcha la paginación de la ROM adicional. Se detecta entonces que es necesaria una rutina para el canal y para ello se utiliza el par de bytes situados cinco bytes más abajo en el área CHANS seleccionada, es decir, se utiliza la rutina de la ROM adicional. Cuando la rutina termina, la ROM deja de ocupar el espacio de memoria que, por paginación, se le había asignado.

El par de bytes de la novena y décima posiciones forman conjuntamente un número de 16 bits que indica al sistema el número de bytes que ocupa el canal en el área CHANS. Este par de bytes son extraordinariamente importantes puesto que cuando hay algún error se utilizan para avanzar por el área CHANS en busca de los canales temporales que es necesario cerrar. Si se omitieran estos bytes o no fueran correctos, el sistema se vería seriamente afectado en cuanto se produjera el primer error.

Utilización de las corrientes

Cuando se quiere emplear una corriente para entrada o para salida hay que recurrir a la rutina STRM_OPEN que está en la posición 1601H (no es en absoluto similar a la instrucción OPEN # del BASIC). Al entrar en la subrutina, el acumulador tiene que contener un número de corriente válido (es decir un número comprendido entre FDH y 0FH). A partir de él se calcula el desplaza-

miento correspondiente que habrá que realizar sobre el área STRMS. Si este desplazamiento es cero se produce el error **Invalid stream** («corriente no válida»). Si no es cero, se calcula la posición correspondiente dentro de CHANS. El valor que allí se encuentre se guarda en la variable del sistema CHURCHL, situada en la posición 5C51H.

Para sacar caracteres por una corriente dada, hay que cargar en el acumulador el código del carácter y luego llamar a la rutina OUTPUT que está en la posición 0010H mediante una instrucción RST 10. Los registros BC, DE y HL no resultan afectados.

Para entrar caracteres desde una corriente dada hay que llamar a la rutina INPUT de la posición 15E6H. Los registros BC, DE y HL permanecerán inalterados. Al retornar de la subrutina el señalizador («flag») de acarreo se activará si el registro A contiene un carácter válido leído. Si no se ha leído carácter alguno, se activará el señalizador de cero. Si se da una situación de fin de fichero tanto el bit de acarreo como el de cero se desactivarán.

Utilización de los canales

Desgraciadamente, no existe un equivalente sencillo en código máquina para la instrucción OPEN #. Para pruebas se podrían abrir las corrientes correspondientes desde el BASIC y utilizarlas desde el código máquina. Probablemente este método no dé resultados muy satisfactorios. Más adelante en este mismo capítulo se dan las equivalencias en lenguaje máquina para cada tipo de instrucción OPEN #. El equivalente en lenguaje máquina de la instrucción CLOSE # es lo siguiente:

```
LD A,corriente
RES 1,(1Y+124)  ;"senalar que no
                  se trata de un
                  CLEAR #"
LD HL,1718H     ;rutina CLOSE de
                 la ROM adicional
LD(HD_11),HL
RST 8
DEFB 32H
RET
```

Se utiliza aquí el código especial 32H para llamar la rutina CLOSE de la ROM adicional.

Rutinas de la ROM

De aquí en adelante se muestran una lista de rutinas de la ROM adicional que sirven para utilizar las posibilidades del Interface. Se dan detalles acerca de cómo utilizar las rutinas (normalmente se emplean códigos especiales) así como información acerca del estado de los registros e interrupciones a la entrada y a la salida de las rutinas. También se indica qué registros quedan afectados por cada rutina. Asimismo, se indica cuál es la posición en la que se encuentran las rutinas.

Rutinas del Microdrive

OPEN_M:	crea un canal «M» temporal en el área CHANS
Para activar la rutina:	RST 8 DEFB 22H
Entrada:	Permitidas las interrupciones
Salida:	IX = principio del área, HL = desplazamiento de la corriente (= IX — CHANS + 1). Permitidas las interrupciones.
Registros:	AF, BC, DE, HL, B"C", D"E". H"L", IX
Posición:	X1B29H/1B05H

Funcionamiento: antes de utilizar esta rutina, la variable del sistema D_STR1 ha de contener el número de la unidad de cartucho, N_STR1 la longitud del nombre del fichero, y T_STR1 tiene que apuntar al principio del nombre del fichero. Primeramente se inserta un bloque de 595 bytes al final del área CHANS (si lo permite el espacio de memoria disponible) y en ella se copian los atributos del fichero correspondientes. Si para la unidad de cartucho es-

pecificada no existe área de mapeo, se crea una que se rellena con FFS. Entonces se pone en marcha la unidad de cartucho y empieza a buscarse el nombre del fichero. Si se encuentra el nombre del fichero, se carga el primer sector y el 0 de CHFLAG se pone a 0 y a 1 para indicar que se trata de un fichero de escritura. Si es un fichero de lectura, el bit 0 de IX + 25 (CHFLAG) se pone a 0 (en caso de ser un fichero PRINT). En otro caso (para programas, bytes, etc.) se deja a 1. Obsérvese que no se comprueba el estado de la lengüeta o pestaña de protección contra escritura. Para realizar esta operación, después de la rutina, hay que hacer lo siguiente:

```
IN A,(EFH)
AND 1            ;Z SI ESTA PROTEGIDO
```

Nótese que esta rutina no desconecta el motor de la unidad de cartucho. El canal que se crea es temporal. Para hacerlo permanente e incorporarlo a una corriente hay que hacer lo siguiente:

```
LD A,corriente
ADD A,A,
LD HL,5C16H
LD E,A
LD D,0
ADD HL,DE
PUSH HL              ;almacenar(save)
                      la posicion de
                      la corriente
RST 8
DEFB OPEN_M
PUSH HL              ;almacenar(save)
                      el desplaza-
                     miento dentro
                      de la corriente
XOR A
RST 8
DEFB MOTOR           ;desconectar
                      motores
POP DE
```

```
POP HL
LD(HL),E
INC HL
LD(HL),D              ;almacenar nue-
                       vos datos en
                       STRMS
RES 7,(IX+4)          ;hacerlo perma-
                       nente
RET
```

MAKE__M: crea un área "M" de 595 bytes en CHANS

Para ponerla en marcha:	**ROM anterior**	**nueva ROM**
	LD HL, 0FE8H	RST 8
	LD (HD__11), HL	DEFB 2BH
	RST 8	
	DEFB 32 H	
Entrada:	Permitidas las interrupciones	
Salida:	IX = inicio del área, HL = IX−CHANS + 1. Permitidas las interrupciones.	
Registros:	AF, BC, DE, HL, B«E», D«E», H«L», IX	
Posición:	X0FE8H/10A5H	

Funcionamiento: Antes de utilizar esta rutina, D__STR1 tendría que guardar el número de la unidad de Microdrive, N__STR1 la longitud del nombre del fichero y T__STR1 su inicio. Se inserta al final de CHANS (si la memoria lo permite) un área de 595 bytes sobre la que se copian los atributos. Si la unidad no tiene ningún mapa, se crea uno y se rellena con FFH. En la ROM anterior no se puede hacer servir el código especial 2BH puesto que, por error, realiza las mismas funciones que el código 22H.

MOTOR: Pone en marcha el motor de una unidad de cartucho o desconecta el de todas.

Para activarla:	RST 8
	DEFB 21H

Entrada:	A debe valer de 1 a 8 para conectar el motor, A = 0 para desconectarlos.
Salida:	Interrupciones prohibidas si A <> 0 y permitidas si A = 0
Registros:	AF, BC, DE, HL
Posición:	X17F7H/1532H

Funcionamiento: Si el acumulador es distinto de cero, el Microdrive correspondiente pondrá su motor en marcha y se desconectarán los motores de los otros Microdrives. Si el acumulador es cero, todos los motores serán desconectados y se permitirán las interrupciones. En el caso de que el Microdrive que se ha seleccionado no estuviera conectado, no existiera cartucho o no estuviera formateado, se produciría un error del tipo Microdrive no presente o inexistente **(Microdrive not present).**

CLOSE__M: cierra un canal "M"

Para activarla:	RST 8 DEFB 23H
Entrada:	IX = inicio del área del canal «M»
Salida:	permitidas las interrupciones
Registros:	AF, BC, DE, HL
Posición:	X12A9/138EH

Funcionamiento: Si el canal escogido es un fichero de escritura, se envían los datos remanentes en la memoria intermedia («buffer»). El motor de la unidad de cartucho se desconecta y se libera el área de 595 bytes. El área de mapeo correspondiente también se libera si no está siendo utilizada por otro canal.

ERASE__M:	borra un fichero guardado en Microdrive
Para activarla:	RST 8 DEFB 24H
Entrada:	nada
Salida:	permitidas las interrupciones

Registros:	AF, BC, DE, HL, B"C", D"E", H"L", IX
Posición:	X1D6EH/1D79H

Funcionamiento: Antes de utilizar la rutina, las variables del sistema D__STR1, T__STR1 y N__STR1 deberían tener por valor el número adecuado de unidad de cartucho. Además debería eliminarse una cadena del fichero. Se crea un canal temporal «M» y se busca el fichero en el cartucho. Si se encuentra, se borra (aunque, en el caso de que haya copias múltiples, sólo resultará borrada una de ellas) sólo si existe la pestaña o lengüeta de protección contra escritura.

Si esta pestaña no existe, se produce un error. Cuando la rutina termina, se desconecta el motor y se elimina el canal temporal.

RECLAIM__M:	Libera un área «M» de 595 bytes
Para activarla:	RST 8 DEFB 2CH
Entrada:	IX = principio del área del canal «M»
Salida:	nada
Registros:	AF, BC, DE, HL
Posición:	X10C4H/119H

Funcionamiento: Primeramente se libera toda el área de 595 bytes y se cierran todas aquellas corrientes que estuvieran apuntando hacia ella. El área de mapeo de 32 bytes se libera únicamente en el caso de que ningún otro canal la esté utilizando.

C AT:	cataloga un cartucho
Para ponerla en marcha:	LD HL,CATANY LD (HD__11), HL RST 8 DEFB 32H • • •
CATANY	LD HL, (04B0H) INC HL

```
LD E, (HL)
INC HL
LD D, (HL)
EX DE, HL
JP (HL)
```

Entrada: Permitidas las interrupciones
Salida: Nada
Registros: AF, BC, DE, HL, A"F", B"C", D"E", H"L", IX
Posición: X1C58H/1C52H

Funcionamiento: Antes de activar esta rutina, S__STR1 debería de contener el número de corriente del catálogo y D__STR1 el número de la unidad de Microdrive. Como no existe ningún código especial para CAT se emplea el código 32H en combinación con HD__11 para llamar a la rutina CATANY cuando la ROM ha sido paginada. CATANY calcula dónde se encuentra la rutina CAT. Para ello recurre al examen de la rutina de sintaxis (que se encuentra en la misma posición en ambas ROMs). Primeramente, la corriente escogida se abre (utilizando 1601H) y luego se crea un canal de tipo «M» temporal. Se pasa a examinar cada uno de los encabezamientos del cartucho y se guarda el nombre del fichero en la memoria intermedia (buffer) de 512 bytes que hay en el área CHANS a menos que ese nombre ya esté allí o empiece por CHR$ 0. Luego, se coloca el nombre en la posición que alfabéticamente le corresponde dentro de la memoria intermedia (buffer). Cuando se ha explorado todo el cartucho o cuando se han encontrado 50 nombres de fichero, se desconecta la unidad de Microdrive y se escribe el nombre del cartucho en la corriente elegida. Después de esto se sacan todos los nombres de fichero que hay en la memoria intermedia (buffer), se imprimen y luego se indica el número de kilobytes libres que aún quedan. Este número se calcula dividiendo por dos la cantidad de bits a cero existentes en el área de mapeo correspondiente. Finalmente, se libera el área «M».

Formato de datos de los cartuchos

Antes de pasar a explicar otras rutinas de mayor complicación, referentes a los Microdrives, es necesario hablar sobre la organiza-

ción de los datos en la cinta. Un cartucho se divide en unos 180 sectores, cada uno de los cuales contiene 512 bytes de datos. En realidad, las rutinas están diseñadas para trabajar con 256 sectores pero los sectores 0 y los que se encuentran por encima del 180 o bien no existen o bien no se usan nunca. Además hay dos o tres sectores que nunca pueden utilizarse por estar situados donde se encuentra la unión de la cinta.

Antes de que se inicie el sector propiamente dicho hay doce bytes «de preámbulo» (diez 0 y 2 FF); luego vienen 15 bytes donde están las variables HDFLAG — HFCHK. Cada encabezamiento tiene un número distinto (HDNUMB) del 1 al 180 almacenado secuencialmente en el cartucho.

Después del encabezamiento viene el sector propiamente dicho. Está formado por 12 bytes de preámbulo (como antes), 15 bytes para las variables (RECFLG a DESCHK), 512 bytes de datos y, finalmente, un byte de Suma de Control (DCHK). Para indicar que un sector no se utiliza, los bits 1 de RECFLG y del byte alto de RECLEN (IX + 70) están desactivados.

Como la mayoría de los ficheros son demasiado grandes para caber en un sector, se subdividen en registros de 512 bytes que se numeran secuencialmente a partir del 0. Si un sector contiene menos de 512 bytes, el bit 1 de RECFLG se activa para indicar que se trata del último sector. Todos los ficheros que no son del tipo PRINT (como los de programas) almacenan sus atributos en los primeros 9 bytes del registro número 0. Estos son los atributos HD__00 a HD__11 que se explican en el Apéndice A.

El resto de códigos especiales que hacen referencia a los Microdrives son:

READ__P:	lee un registro de un fichero PRINT
Para activarla:	RST 8
	DEFB 27H
Entrada:	IX = inicio del área del canal «M»
Salida:	Permitidas las interrupciones, motor en marcha
Registros:	AF, BC, DE, HL
Posición:	X1A17H/1F0BH

Funcionamiento: Antes de empezar a utilizar esta rutina, CHDRIV debería contener el número de la unidad de cartucho, CHRC el número de registro requerido y CHNAME el nombre del fichero. Primeramente se pone en marcha la unidad de cartucho escogida y a SECTOR se le da el valor 04FB (= cinco revoluciones completas). Se busca entonces en el cartucho, dentro del fichero escogido, el número del registro indicado. Si se encuentra y el fichero es del tipo PRINT, se transfiere el sector a la memoria intermedia (buffer) y se retorna de la rutina. Si el fichero no es de tipo PRINT, entonces se libera el área y se produce un error Tipo de fichero equivocado **(Wrong file type).** Si después de cinco búsquedas consecutivas no se ha conseguido encontrar el registro buscado, se producirá el error Fichero no encontrado **(File not found)** y, si era temporal, se liberará el área.

READ_NP:	lee el siguiente registro de un fichero de tipo PRINT
Para activarla:	RST 8 DEFB 25H
Entrada:	IX = inicio del área «M»
Salida:	Prohibidas las interrupciones, motor en marcha
Registros:	AF, BC, DE, HL
Posición:	X1A09H/1EFDH

Funcionamiento. Esta rutina, similar a READ_P y a CHDRIV, debería de tener por valor el número de la unidad de cartucho y CHNAME el nombre del fichero. Primeramente, se inspecciona el bit 1 de RECFLG para ver si el registro que está en ese momento en memoria es el último o no. Si lo es, se producirá un error de **Fin de fichero.** En caso contrario, se incrementará el contenido de CHREC y el control pasará a READ_P.

READ_S:	lee el sector siguiente
Para activarla:	RST 8 DEFB 29H

Entrada: IX = inicio del área "M" prohibidas las interrupciones, motor en marcha.

Salida: prohibidas las interrupciones, motor en marcha

Registros: AF, BC, DE, HL

Posición: X1A86H/1F7AH

Funcionamiento: El encabezamiento inmediato y el sector son leídos en el área del canal en la cinta. Si se trata de un fichero del tipo PRINT se realiza un retorno de la subrutina y se desactiva el señalizador de acarreo. Si no se trata de un fichero del tipo PRINT o falla el control de suma, se borra el contenido de la memoria intermedia (buffer) y el retorno se realiza con el señalizador de acarreo activado.

READ__R: lee un sector de un fichero PRINT directamente

Para activarla: RST 8
DEFB 28H

Entrada: IX = inicio del área «M» prohibidas las interrupciones, motor en marcha

Salida: prohibidas las interrupciones, motor en marcha

Registros: AF, BC, DE, HL

Posición: X1A4BH/1F3FH

Funcionamiento: Se busca en el cartucho el sector cuyo número es CHRC. Si se encuentra, se transfiere a la memoria intermedia (buffer). Si el fichero es de tipo PRINT, se pone a cero el señalizador de acarreo y se retorna de la subrutina. Si no es un fichero del tipo PRINT, se borra el contenido de la memoria intermedia (buffer) y se activa el señalizador de acarreo. Si, después de una vuelta, no se ha encontrado el sector buscado, se produce un error Fichero no encontrado **(File not found).**

SCAN__M: lee un encabezamiento del cartucho (sólo en ROM nueva)

Para activarla:	RST 8 DEFB 33H
Entrada:	IX = inicio del área «M», prohibidas las interrupciones, motor en marcha
Salida:	prohibidas las interrupciones, motor en marcha
Registros:	AF, BC, DE, HL
Posición:	X1FE4 (sólo en la ROM nueva)

Funcionamiento: Se lee la cinta que pasa ante la cabeza del Microdrive hasta encontrar un encabezamiento válido que se transferirá a la zona comprendida entre las posiciones HDFLAG y DESCHK ambas inclusive. Si la Suma de Control no concuerda o si el nombre de fichero del sector empieza con CHR$ 0, se borra el contenido de toda el área y se activa el bit de acarreo. En cualquier otro caso, se desactiva el bit de acarreo. Para leer los encabezamientos de sectores adyacentes es necesario que cualquier código que se ejecute, entre dos llamadas consecutivas, tarde menos de 30 ms.

WRITE_S:	escribe un sector secuencialmente
Para activarla:	RST 8 DEFB 26H
Entrada:	IX = inicio del área «M»
Salida:	Permitidas las interrupciones, motor parado
Registros:	AF, BC, DE, HL
Posición:	X11FFHH/12DAH

Funcionamiento: Antes de la llamada a la subrutina, las variables CHBYTE, CHREC, CHNAME y CHDRIVE así como la memoria intermedia (buffer) deben estar inicializados convenientemente. Primeramente, se pone en marcha el motor correspondiente y se comprueba todo el cartucho. Se transfiere el nombre del fichero de CHNAME a RECNAME y CHBYTE pasa a CHREC así como RECLEN a RECNUM. Se calculan las sumas de control (checksums) DESCHK y DCHK. El área comprendida entre RECFLG y DCHK se escribe en el primer sector libre del cartucho (en el supuesto de que no esté

protegido contra la escritura). Se da el valor adecuado al bit correspondiente del área de mapeo. Se da el valor cero a CHBYTE y se incrementa CHREC. Finalmente, se desconecta el motor.

WRITE_R:	escribe un sector directamente.
Para activarla:	RST 8 DEFB 2AH
Entrada:	IX = inicio del área «M», las interrupciones prohibidas, el motor en marcha
Salida:	prohibidas las interrupciones, motor en marcha
Registros:	AF, BC, DE, HL
Posición:	X1A91H/1F85H

Funcionamiento: Antes de llamar a CHREC ésta debería tener el valor que se desearía atribuirle, y CHNAME debería contener el nombre del sector que tendría que ser conservado. Se busca una sola vez en el cartucho el sector cuyo número es el CHREC. Si no se puede encontrar, se genera el error Fichero no encontrado **(File not found)**. Si se encuentra en el cartucho, se comprueba la lengüeta o pestaña de protección contra escritura. Si existe, se escriben en el cartucho los bytes del sector y de datos (RECFLG—DCHK) y se activa el bit correspondiente en el área de mapeo. Obsérvese que antes de escribir el área de mapeo no se realiza ninguna comprobación acerca de su contenido. En todo caso, debe ser el usuario quien, si lo desea, la realice.

Rutinas del RS232

Antes de utilizar cualquier rutina relacionada con el RS232 deben existir las variables adicionales del sistema y se ha de indicar la velocidad en baudios escogida. Lo primero puede conseguirse mediante el código especial 31H y lo segundo dando el valor necesario a BAUD (posición 5CC3H).

232IN:	lee un byte del RS232
Para activarla:	RST 8 DEFB 1DH

Entrada:	nada
Salida:	el señalizador de acarreo estará activado si se ha leído un byte; A = byte leído, permitidas las interrupciones.
Registros:	AF, BC, DE, HL
Posición:	X0B81H/0B88H

Funcionamiento: La rutina leerá un byte de la vía de acceso (port) del RS232 a menos que el tiempo que deba esperar sea demasiado largo o se pulse la tecla SPACE (ROM anterior) a BREAK (ROM nueva). En este último caso, puede producirse una ruptura de programa **(Break into program).** Si se ha leído un byte, el señalizador de acarreo quedará activado.

232OUT: envía un byte a través del RS232

Para activarla:	RST 8 DEFB 1EH
Entrada:	Permitidas las interrupciones
Salida:	A = código del byte
Registros:	AF, BC, DE, HL
Posición:	X0C5AH/0D07H

Funcionamiento: Se envía un byte a través de la vía de acceso (port) con la velocidad correcta y siguiendo el protocolo necesario. Si se pulsa BREAK durante la transmisión, se producirá un error.

Si se quiere utilizar el equivalente del canal «T» para realizar una salida a través del RS232, hay que emplear el código especial 32H utilizado normalmente para llamar a X0C3CH (ROM anterior) o X0C3AH (ROM nueva). A debe contener el código del carácter que se desee enviar. La nueva ROM utiliza dos variables más del sistema cuando maneja canales «T»: 5CB0H para la posición del cursor y 5CB1 para la anchura de la impresora, inicializándose a 0 y 80, respectivamente.

OPEN _ R:	abre un canal del RS232	
Para activarla:	**antigua ROM**	**ROM nueva**
	LD HL, 0B13H	LD HL,0B17H
	LD (HD__11), HL	LD (HD__11), HL
	RST 8	RST 8
Entrada:	Permitidas las interrupciones	
Salida:	DE = inicio de la nueva área CHANS	
Registros:	AF, BC, DE, HL	
Posición:	X0B13H/0B17H	

Funcionamiento: Esta rutina crea un área de 11 bytes al final de CHANS para un canal «B» o «T». Normalmente creará un canal de tipo «T». En caso de que L__STR1 contenga una «B», se creará un canal de tipo «B». En la nueva ROM hay un código especial nuevo llamado OPEN__B,34H que abre un canal «B».

Rutinas de la red

El establecimiento de la red en el Spectrum tiene lugar mediante una continua conexión entre las vías de acceso (ports) de todas las estaciones. Sólo se establece una comunicación cada vez y todos los datos se transmiten en serie. Los datos se subdividen en «paquetes» numerados. La longitud máxima de cada paquete es de 255 bytes. Cada paquete se transmite de la siguiente manera: primero, el emisor comprueba si la red ya está ocupada y, en tal caso, espera. Cuando la red está libre, el emisor envía un encabezamiento compuesto de 8 bytes (las variables del sistema NTDEST y NTCHS). En este encabezamiento se guarda información acerca de los datos que vendrán a continuación. Entre otras cosas vendrán los números del emisor y del receptor y dos sumas de control. A menos que esté realizando una emisión, la estación que envía el mensaje comprueba si le ha llegado una señal de reconocimiento del receptor. Esta señal es un solo byte que contiene el número de la estación receptora. Si no se recibe este byte, se vuelve a transmitir el encabezamiento. En el caso de realizarse una emisión, no se espera la llegada de ese byte de reconocimiento y se envían los datos directamente. Luego se envía la sección que contiene los datos en sí (a menos que no haya ningún byte en ella, es decir,

que NTLEN = 0). Hecho esto, se comprueba si se recibe un nuevo byte de reconocimiento (a menos que se trate de una difusión). Si este byte no se recibe, se repite el proceso volviéndose a transmitir el encabezamiento. Hay cuatro códigos especiales para emplear la red aunque, desgraciadamente, uno de ellos no es utilizable en la anterior ROM.

OPEN__N:	abre un canal temporal «N»
Para activarla:	RST 8 DEFB 2DH
Entrada:	nada
Salida:	IX = DE = (CURCHL) = inicio del área "N" dentro de CHANS
Registros:	AF, BC, DE, HL
Posición:	X0EA9H/0F46H

Funcionamiento: Antes de llamar a esta subrutina D__STR1 tiene que contener el número del destinatario y NSTAT el del Spectrum. Se crea al final del área CHANS un área de 256 bytes donde se almacenan los datos. El número del destinatario se pasa de D__STR1 a NCIRIS, el número de la estación emisora de NSTAT a NCSELF y el área que va de NCNUMB al último carácter de la memoria intermedia (buffer) se rellena con ceros. El canal se hace temporal estableciendo el bit 7 del indicador de canal. Se hace que la variable CURCHL apunte al principio del área que se acaba de crear.

Para crear un canal permanente de la red se puede emplear el siguiente código (después de dar los valores adecuados a D__STR1 y NSTAT).

```
LD A,corriente
ADD A,A
LD HL,5C16H
LD E,A
LD D,0
ADD HL,DE          ;HL=posicion ade
                    cuada dentro de
                    STRMS
PUSH HL            ;almacenarla
RST 8
```

```
DEFB OPEN_N          ;abrir un "N"
                      temporal
RES 7,(IX+4)         ;hacerlo perma-
                      nente
LD HL,(CHANS)
EX DE,HL
AND A
SBC HL,DE
INC HL               ;HL=desplazamien
                      to dentro de la
                      corriente
POP DE               ;DE=posicion de
                      STRMS
EX DE,HL
LD(HL),E             ;almacenar el
                      desplazamiento
                      en STRMS
INC HL
LD(HL),D
RET
```

CLOSE_N:	cierra un canal «N»
Para activarla:	RST 8 DEFB 2EH
Entrada:	(CURCHL): inicio del área «N» dentro de CHANS
Salida:	Permitidas las interrupciones
Registros:	AF, BC, DE, HL, IX
Posición:	X1A24H/1F18H

Funcionamiento: Si el canal es un fichero de escritura (por ej., NCOBL < > 0) entonces el resto del contenido de la memoria intermedia (buffer) se envía como un paquete marcado con EOF. Se libera entonces el área de 256 bytes. En el caso de que hubiera alguna corriente asignada al canal **no** será cerrada. El área «N» tiene que ser o bien temporal o bien la última área que hay en CHANS pues, de otra forma, pueden quedar perjudicadas otras corrientes y canales.

WRITE_N: envía un paquete a través de la red

Para activarla: RST 8
DEFB 30H

Entrada: A=0 para datos, 1 para EOF, IX= inicio del área "N"

Salida: Permitidas las interrupciones, A = número del destinatario (Z si se trata de una difusión)

Registros: AF, BC, DE, HL

Posición: X0DB2H/0E4FH

Funcionamiento: Antes de llamar a la subrutina, las variables NCOBL y NCNUMB, el contenido de la memoria intermedia (buffer) y el del encabezamiento (excluidas las sumas de control «checksums») deben tener los valores adecuados. Al entrar en la subrutina, el acumulador contendrá normalmente el valor 0 o bien un 1 si el paquete es el último de su serie (es decir un indicador de fin de fichero). Este valor se almacena en NCTYPE y el color del borde de la pantalla pasa a ser el indicado por IOBORD. Se calcula la suma de control («checksum») de la información contenida en la memoria intermedia (buffer) y el valor obtenido se guarda en NCDCS. Igualmente, se pasa a calcular la suma de control ("checksum") de la información contenida entre NCIRIS y NCDCS guardándose en NCHCS. Se impiden las interrupciones y, luego, se envía el paquete. Cuando se ha recibido el paquete (si no se trata de una difusión) se pasa a incrementar el contenido de NCNUMB. Luego, se hace que el color del borde de la pantalla vuelva a ser el mismo de antes. Finalmente, se permiten las interrupciones.

READ_N: Su supuesta misión es leer un paquete de la red.

Para activarla: RST 8
DEFB 2FH

Entrada: IX = inicio del área «N»

Salida: permitidas las interrupciones

Registros: AF, BC, DE, HL

Posición: X1A31H/1F25H

Funcionamiento: Este código especial lee un paquete de bytes de la red. El señalizador de acarreo se desactiva si la lectura del

paquete se ha realizado correctamente. En caso contrario, se activa, pero, en la ROM antigua, se pierde el contenido de este señalizador.

La mejor forma de leer los caracteres de la red, como ya se dijo, es a través de la rutina INPUT (contenida en la ROM de 16 K), situada en la posición 15E6H. Recuerde conservar el valor de CURCHL si emplea cualquier otro canal entre llamadas sucesivas a esta rutina.

Códigos especiales de objetivo diverso

PAUSE:	espera a que se pulse (o repita) una determinada tecla
Para activarla:	RST 8 DEFB 1BH
Entrada:	nada
Salida:	Permitidas las interrupciones, A = código de la tecla
Registros:	AF, BC, DE, HL
Posición:	X19D9H/1ECDH

Funcionamiento: Se permiten las interrupciones y cada 1/50 de segundo se llama a la rutina de exploración del teclado que está grabada en la ROM de 16 K. Este proceso se repite hasta que se pulsa una tecla. El valor de la tecla se extrae de LAST_K.

PRINT:	Imprime un carácter en la pantalla.
Para activarla:	RST 8 DEFB 1CH
Entrada:	A = código del carácter, permitidas las interrupciones
Salida:	nada
Registros:	AF, BC, DE, HL, A''F', B''C'', D''E''
Posición:	X19ECH/1EE0H

Funcionamiento: se da el valor FF a la variable SCR_CT para permitir el desplazamiento automático de la pantalla («scrolling»). Se abre la corriente FEH (canal «S») y, finalmente, se imprime el carácter.

LPRINT: imprime un carácter en la impresora ZX
Para activarla: RST 8
DEFB 1FH
Entrada: A = código del carácter, permitidas las interrupciones
Salida: nada
Registros: AF, BC, DE, HL, A"F", B"C", D"E"
Posición: X19FCH/1EF0H

Funcionamiento: La corriente 3 (normalmente el canal «P») se abre y se imprime el carácter.

SCAN_K: explora la matriz del teclado
Para activarla: RST 8
DEFB 20H
Entrada: nada
Salida: NZ si se ha pulsado alguna tecla
Registro: AF, BC, DE, HL
Posición: X1A01H/1EF5H

Funcionamiento: Se explora directamente el teclado y si se ha pulsado alguna tecla (incluidas las que se activan mediante SHIFT) se desactiva el señalizador Z. Las interrupciones no son necesarias.

NEWVARS: crea las variables adicionales del sistema
Para activarla: RST 8
DEFB 31H
Entrada: nada
Salida: nada
Registros: AF, BC, DE, HL
Posición: X19A8H/1E98H

Funcionamiento: si no existen aún las 58 variables del sistema adicionales (ver Apéndice A), se crean (si hay suficiente memoria). (De hecho la rutina tan sólo contiene la instrucción RET; la propia instrucción RST 8 crea las variables adicionales).

SHADOW:	llama a cualquier rutina de la ROM adicional
Para activarla:	RST 8 DEFB 32H
Entrada:	(HD__11) = posición
Salida:	nada
Registros:	sin especificar (depende de la rutina a que se llame)
Posición:	X19A4H/1E94H

Funcionamiento: Se llama a la rutina a la que apunta HD__11. Aunque oficialmente reciba el calificativo de **Reserved by Sinclair Research Ltd.** (Reservado para futura ampliación por SRL (Sinclair Research Laboratories) éste es el código especial de mayor utilidad de la ROM. Sólo se puede pasar a la subrutina el valor contenido en el registro A pero, en cambio, devuelve todos los valores. Es posible emplear este código especial, incluso, para terminar la paginación del BASIC del ROM. Esto se consigue con el siguiente código:

```
       :LD HL,PAGOUT
       :LD (HD_11),HL
       :RST 8
       :DEFB 32H
PAGOUT:POP HL         ;sacar 0700H
                       de la pila
       :POP HL        ;sacar PAGOUT
                       de la pila
          "           ;resto del
                       programa
          "
```

Resumen de los códigos especiales

En este resumen se lista cada código especial, su posición dentro de la ROM adicional así como la función que realiza.

Código	Nombre	Posición	Función
1B	PAUSE	19D9/1ECD	Espera a que se pulse una tecla. El valor está en A

Código	Nombre	Posición	Función
1C	PRINT	19EC/1EE0	imprime CHR$ A en la corriente FE (la pantalla)
1D	232 IN	0B81/0B88	La entrada que viene del RS 232 va al registro A. Se activa el acarreo si el carácter es válido
1E	232OUT	0C5A/0D07	Salida por RS232. Código del carácter en A
1F	LPRINT	19FC/1EF0	Imprime CHR$ A en la corriente 3 (la impresora)
20	SCAN__K	1A01/1EF5	Explora la matriz del teclado (NZ si se pulsa cualquier tecla)
21	MOTOR	17F7/1532	Conecta o desconecta los motores de los Microdrives
22	OPEN__M	1B29/1B05	Abre un canal «M» temporal
23	CLOSE__M	12A9/138E	Cierra un canal «M»
24	ERASE__M	1D6E/1D79	Borra un fichero del cartucho
25	READ__NP	1A09/1EFD	Lee la siguiente memoria intermedia (buffer), de tipo PRINT
26	WRITE__S	11FF/12DA	Envía el contenido de la memoria intermedia (buffer) «M» al cartucho
27	READ__P	1A17/1F0B	Lee la memoria intermedia (buffer) de tipo PRINT
28	READ__R	1A4B/1F3F	Búsqueda en un fichero directo
29	READ__S	1A86/1F7A	Búsqueda en un fichero secuencial
2A	WRITE__R	1A91/1F85	Escribe un bloque directamente
*2B	MAKE__M	— —/10A5	crea un área de tipo «M»
2C	RECLAIM	10C4/119F	Libera un área «M» de 595 bytes
2D	OPEN__N	0EA9/0F46	Abre un canal «N» temporal
2E	CLOSE__N	1A24/1F18	Cierra un canal «N»

Código	Nombre	Posición	Función
*2F	READ_N	– –/1F25	lee un paquete de la red
30	WRITE_N	0DB2/0E4F	Escribe un paquete de la red
31	NEWVARS	19A8/1E98	Crea nuevas variables del sistema si es necesario
32	SHADOW	19A4/1E94	Llama a la rutina de la ROM adicional direccionada por HD_11
*33	SCAN_M	– –/1FE4	lee el siguiente encabezamiento de cartucho
*34	OPEN_B	– – / 1FF6	abre un canal «B»

* significa «únicamente para la nueva ROM»

Notas

Cuando se programa en código máquina con el Interface hay que tener en cuenta una serie de consideraciones:

i) La función USR sólo se debe utilizar dentro de sentencias LET o RANDOMIZE. Si se utiliza en las sentencias del BASIC extendido, y se produce un error, la ROM adicional puede afectar fatalmente al sistema.

ii) Hay que tener muy en cuenta el estado de las interrupciones al entrar o salir de ciertas rutinas de la ROM adicional. Antes de volver al BASIC deben PERMITIRSE las interrupciones.

iii) Si se utilizan las rutinas del Microdrive hay que conservar el valor del registro H"L" si se va a volver al BASIC.

iv) Nunca hay que utilizar cartuchos importantes cuando se esté procediendo a la depuración de rutinas de Microdrive. Puede resultar bastante molesto formatear accidentalmente el cartucho que contenga todo nuestro código original. Las lengüetas de protección contra escritura no garantizan una seguridad completa ante errores de código máquina graves.

v) Hay que asegurarse de que se han creado las variables adicionales del sistema con el código 31H.

vi) No hay que utilizar sentencias REM en las rutinas del Interface. La creación de áreas CHANS las desplazaría por el mapa de memoria afectando fatalmente al sistema.

Capítulo 9

INCLUSION DE NUEVAS INSTRUCCIONES EN BASIC

Así como el Interface 1 añade nuevos comandos al BASIC extendido del Spectrum, también el usuario puede añadir los suyos. Esta es una posibilidad muy interesante que se encuentra en muchos otros ordenadores pero que siempre había faltado en los de la gama Sinclair. La razón principal que explica esta omisión es que, hasta ahora, todos los comandos del Spectrum tenían que ser palabras clave. Otra razón es que no había espacio libre suficiente en el conjunto de caracteres del Spectrum para añadir nuevos comandos. Lo que hace el Interface 1 es aportar la opción de saltar a un programa adicional en código máquina si se encuentra un error de sintaxis en una línea ya sea en el momento de introducirla ya sea en el momento de ejecutarla.

De esta forma, se puede cambiar la sintaxis de la mayoría de los comandos existentes para que realicen otra función a pesar de que los comandos del Interface no pueden cambiarse. Existen varios métodos de añadir nuevos comandos que no sean del Interface.

1) Insertar caracteres en lugares extraños, por ejemplo **COPY #**
2) cambiar el tipo de los parámetros, por ejemplo, **POKE 30000, «x»**
3) añadir parámetros, por ejemplo, **PLOT x,y, "BANG"**
4) eliminar parámetros, por ejemplo, **CIRCLE 10,30**
5) utilizar funciones del modo E, por ejemplo, **LINE 10,30**

El usuario puede también utilizar sus propios comandos con todas sus letras siempre que vayan precedidas de un carácter no alfa-

bético (por ejemplo **!REPETIR**). Es necesario colocar ese carácter no alfabético para que, cuando se introduzca esa línea, el Spectrum salga del modo K.

Las palabras clave cuya sintaxis normalmente no se puede alterar son LOAD, SAVE, MERGE, VERIFY, CAT, FORMAT, OPEN, CLOSE, CLS, CLEAR y MOVE. Digo «normalmente» porque, de hecho, se pueden alterar si a usted no le importa realizar un poco de gimnasia con los dedos cuando las teclee. La forma de modificarlas es precediéndolas de un carácter que requiera el empleo de la tecla SHIFT, por ejemplo, "*". Esto significa que las palabras clave deben introducirse tecleando primero la palabra, haciendo luego «cursor hacia la izquierda», tecleando después "*", haciendo «cursor hacia la derecha» y luego introduciendo el resto de la línea.

Para crear sus propios comandos debe usted tener buenos conocimientos del lenguaje máquina y del sistema operativo del Spectrum. En la segunda parte de este capítulo se explican los métodos a utilizar por parte de los lectores que ya posean ese nivel de conocimientos. La primera parte del capítulo contiene un programa que altera la sintaxis de los comandos de Microdrive más corrientes, programa que puede ser introducido y utilizado por cualquier usuario.

Para indicarle al Interface que se han añadido nuevos comandos a los ya existentes se tienen que hacer dos POKEs que **necesariamente** deben ir en una línea múltiple, es decir, de varias sentencias. No debería hacerse esto cuando, después de haber hecho NEW, no se ha efectuado aún ninguna operación del Interface. Lo más sencillo es hacer **CLOSE #** que, en realidad, es una instrucción sin efecto. Para hacer que el Interface vuelva a trabajar únicamente con comandos normales, se necesita:

```
POKE 23735,240: POKE 23736,1
```

igualmente en una sola línea.

Tengo que admitir que no me gusta demasiado la sintaxis utilizada para almacenar y cargar programas en o desde un Microdrive. Bajo mi punto de vista, se trata de una sintaxis innecesariamente complicada que requiere una cantidad excesiva de pulsaciones. Por este motivo escribí el siguiente programa que añade nuevos comandos para realizar las mismas operaciones. Todos ellos tienen el mismo

formato. Después de un asterisco se escribe una letra (mayúscula o minúscula) que identifica al comando correspondiente y después se añaden algunos detalles. Para cargar un programa llamado «test» desde la unidad de cartucho número 2 el nuevo comando sería *L"2test" que equivale a LOAD*"m"; 2;"test". De forma similar, SAVE*"m",VERIFY y MERGE se sustituyen por *S, *V y *M seguidas de una sola cadena. La primera letra de ésta tiene que ser el número de la unidad de cartucho. Otro de los comandos que se añaden es la *C que equivale a CAT 1. Si después de *C se coloca otro número, se pueden obtener los catálogos de las otras unidades. El último comando que he añadido es la ***run** con la que se carga desde la unidad 1 un programa llamado «run». Es equivalente a **NEW** seguido de RUN.

Listado de los comandos adicionales

```
  10 REM ************************
  20 REM * AMPLIAR LA SINTAXIS *
         *   DEL MICRODRIVE     *
  30 REM ************************
  40 REM                PARA 16K
  50 CLEAR 65169:   REM 32401
  60 LET ex=65170: REM 32402
  65 LET x2=INT ((ex+124)/256):
LET x1=ex+124-256*x2
  70 RESTORE 200: LET c=0
  80 FOR i=ex TO ex+194
  90 READ a: LET c=c+a
 100 POKE i,a
 110 NEXT i
 115 IF c<>22003+2*(x1+x2) THEN
 PRINT "ERROR EN SUMA DE CONTROL
": STOP
 120 CLOSE #0
 130 POKE 23735,ex-256*INT (ex/2
56): POKE 23736,INT (ex/256)
 140 PRINT "NUEVAS INSTRUCCIONES
:"
 150 PRINT "*L LOAD"'"*S SAVE"'"
*M MERGE"'"*V VERIFY"
```

```
 160 PRINT "*C CAT"'"*RUN"
 170 STOP
 200 DATA 254,92,194,240,1,215,3
2,0
 210 DATA 246,32,254,115,40,22,2
54,108
 220 DATA 40,28,254,118,40,34,25
4,109
 230 DATA 40,36,254,99,40,38,254
,114
 240 DATA 40,62,24,222,253,203,1
24,238
 250 DATA 205,x1,x2,195,54,8,253
,203
 260 DATA 124,230,205,x1,x2,195,
165,8
 270 DATA 253,203,124,254,24,244
,253,203
 280 DATA 124,246,24,238,33,214,
92,54
 290 DATA 1,35,54,0,35,54,2,215
 300 DATA 32,0,254,13,40,7,254,5
8
 310 DATA 40,3,205,30,6,195,169,
4
 320 DATA 215,32,0,246,32,254,11
7,194
 330 DATA 40,0,215,32,0,246,32,2
54
 340 DATA 110,32,244,215,32,0,20
5,183
 350 DATA 5,195,149,10,62,77,50,
217
 360 DATA 92,215,32,0,215,140,28
,215
 370 DATA 24,0,223,202,35,7,215,
241
 380 DATA 43,33,11,0,167,237,66,
218
 390 DATA 76,6,120,177,202,76,6,
26
```

```
400 DATA 214,48,254,1,56,27,254
,9
410 DATA 48,23,50,214,92,175,50
,215
420 DATA 92,11,19,237,67,218,92
,237
430 DATA 83,220,92,215,24,0,195
,35
440 DATA 7,231,4
```

Creo que estos nuevos comandos son mucho más fáciles y rápidos de utilizar y he guardado el programa bajo el nombre de «run» en el cartucho que utilizo principalmente para desarrollo de programas. Cuando se introduzca este programa habrá que cambiar las líneas 50 y 60 para que cambien la nueva posición de RAMTOP y la posición de memoria donde debe ir el programa. Las posiciones recomendadas para este programa son incompatibles con las que se han recomendado para otras rutinas de este mismo libro. Si se desea que sean compatibles habrá que añadir las líneas:

```
50 CLEAR 64579: REM 31809 PARA
16K
60 LET mc=64580: REM 31810 PAR
A 16K
```

Haciendo NEW se impedirá la utilización de los comandos adicionales recién creados por el usuario. Por tanto, es necesario preparar un fichero de programas vacío con el nombre de «new». Entonces, para borrar el programa en ejecución mientras se mantiene la validez de los nuevos comandos será necesario hacer:

```
*L"1NEW"
```

Cómo se añaden nuevas instrucciones

Esta parte del capítulo está dirigido a los programadores en código máquina. Resulta útil disponer de un cierto conocimiento del sistema operativo del Spectrum (con este fin recomiendo el libro

del Dr. Ian Logan «The Complete Spectrum ROM Disassembly» cuyas etiquetas de las rutinas de la ROM de 16 K he utilizado aquí).

La variable VECTOR de la posición 5CB7H es vital para utilizar los comandos creados por el usuario. Normalmente esta variable contiene el valor 01F0H que es una posición de la ROM adicional que ejecuta la rutina normal de gestión de errores (después de copiar de nuevo CHADD_ sobre CH_ADD).

Para explicar cómo se puede alterar la variable VECTOR hay que conocer primero las acciones que realiza la ROM adicional cuando se produce un error. Los números entre paréntesis se refieren a las posiciones de la ROM en hexadecimal.

Primero, se hace que el par de registro HL contenga la dirección de retorno (0013) (desde RST 8) y se comprueba si es 0000 ó 15FE. Si es 0000 significa que se ha llamado desde la ROM adicional una rutina que está en los 16 K de ROM normales. Si es 15FE significa que se ha pedido un canal del Interface. Si no es ninguno de esos dos valores, se crean las variables del Interface (si aún no se habían creado) y se envían algunas señales a la ULA que hay en el interior del Interface. Se ve entonces cuál es el número del error (00C7) (del byte que sigue al RST 8) y se comprueba si se trata de un código especial o no. En el caso de que no se trate de un código especial, se comprueba si el error es del tipo Sin sentido en BASIC **(Nonsense in BASIC)**, Nombre de fichero incorrecto **(Invalid filename)** o Corriente incorrecta **(Invalid stream)**. Si el error no era ninguno de estos tipos, entonces se utiliza la rutina de gestión de errores guardada en la ROM original (00F8). En el caso de que se tratara de alguno de los errores anteriormente mencionados se copia el contenido de CH_ADD sobre CHADD_ (¡Vaya mnemónicos tan propensos a la confusión que se han escogido!). Se comprueba el estado del bit 3 de FLAGS3 y, si está activado, se recurre a la rutina normal de gestión de errores. Después de calcular cuál es el primer carácter de la línea en que se ha producido el error (0126-01E9) se compara con las palabras especiales del Interface (01B5-01E9). Si no se trata de una palabra especial, se salta a la rutina a la que apunta la variable VECTOR. Normalmente, se produce un error al hacer esto pero, si se cambia el contenido de VECTOR, se pueden añadir comandos que superen todos los pasos de comprobación de la corrección de la sintaxis.

Por tanto, si se ha cambiado el contenido de VECTOR para que apunte a una de nuestras rutinas adicionales ¿qué debe hacer ésta? Primeramente debe comprobar cuál es el contenido del registro A. Hay que ver si contiene la primera palabra especial o código de carácter del comando adicional creado por el usuario (tiene que ser menor que 206). Si no se reconoce el contenido de A como indicador de una de esas dos posibilidades, hay que saltar a la posición 01F0H (JP 01F0H) para que se produzca un error. En caso de que se haya producido un reconocimiento positivo hay que saltar a la rutina que se encarga de comprobar la sintaxis del comando, evaluar los parámetros que contenga y ejecutarlo (esto último únicamente en tiempo de ejecución). Hay que tener presente un par de cosas al escribir este tipo de rutinas:

1. Por paginación, ocupa el espacio de memoria la ROM adicional y no los 16 K del BASIC.
2. Todos los «restarts» del Z80 son distintos.
3. **NO** hay que intentar emplear los códigos especiales: hay que llamar a las rutinas directamente.
4. Incluso cuando se permiten las interrupciones, **no** se explora el teclado ni se incrementa el contenido de FRAMES.

Los distintos RST («restarts») de la ROM adicional hacen lo siguiente:

RST 0: desactiva FLAGS3 y retorna a la ROM de 16 K

RST 8: **no utilizarlo nunca.**

RST 10: llama a las rutinas de la ROM de 16 K. Está seguido de los dos bytes de datos con que se identifica la posición de la rutina.

RST 19: hace BIT 7, (FLAGS), da Z si se está realizando una comprobación de sintaxis y NZ si se está en ejecución.

RST 20: realiza un error de la ROM adicional. Va seguido de uno de los siguientes bytes:

FF	Programa terminado	00E7
00	Sin sentido en BASIC	0139
01	Número de corriente incorrecto	0663

02	Expresión incorrecta de dispositivo	062D
03	Nombre incorrecto	064C
04	Número incorrecto de unidad de cartucho	0681
05	Número incorrecto de estación	05F6
06	Falta nombre	068D
07	Falta el número de la estación	06A1
08	Falta el número de la unidad de cartucho	0683
09	Falta especificar la velocidad en baudios	06B7
0A	Error de concordancia en el encabezamiento	(nunca se utiliza en la ROM)
0B	Corriente ya abierta	052F
0C	Escritura sobre un fichero de lectura	0D78
0D	Lectura de un fichero de escritura	0D1C
0E	Unidad de cartucho protegida contra escritura	128D
0F	Microdrive lleno	1219
10	Microdrive inexistente	1828
11	Fichero no encontrado	11A3
12	Código de error especial	1985
13	Error de CODE	092E
14	Error de MERGE	07D8
15	La verificación ha fallado	0930
16	Tipo incorrecto de fichero	0902

(las direcciones en hexadecimal son a las que se debe saltar para producir cada uno de los errores reseñados).

RST 28: realizar un error de la ROM de 16 K. Cargar la variable ERRNR con el código de mensaje de error correspondiente antes de realizar el RST.

RST 30: crear las variables del Interface si aún no existen.

RST 38: permitidas las interrupciones (por esta razón no se explora el teclado).

Rutinas de exploración de líneas

Cualquiera que sea el comando que se añada, será necesario explorar una línea de BASIC ya sea para ejecutar el comando o

para comprobar su sintaxis. La rutina creada por el usuario será llamada dos veces por cada una de las líneas entradas: una cuando se compruebe la sintaxis y otra cuando se proceda a la ejecución. Para saber en qué paso se está hay que utilizar RST 18.

En la ROM original hay muchas rutinas de exploración de líneas que pueden ir perfectamente para este fin. Se utilizan mediante RST 10. Para explorar la línea carácter a carácter se deben utilizar las siguientes rutinas de la ROM original:

GET_CHAR 0018H:	el registro A contiene el byte que se está tratando dentro de una línea.
NEXT_CHAR 0020H:	El registro A contiene el siguiente byte de la línea (se obtiene el siguiente byte prescindiendo de códigos de control y de los espacios)

Las rutinas principales para realizar una evaluación son:

CLASS_061C82H:	comprueba/evalúa un parámetro numérico y pone el valor en la pila del calculador si se está en tiempo de ejecución; en otro caso, inserta en la línea algunos bytes invisibles.
CLASS_0A1C8CH:	comprueba/evalúa un parámetro de tipo cadena y pone el valor en la pila del calculador si se está en tiempo de ejecución.

Al entrar en las dos últimas rutinas CH_ADD tiene que apuntar al primer carácter de la expresión. Al retorno de la subrutina CH_ADD apunta al primer carácter «fuera de sitio». El registro A contiene el código correspondiente a este carácter (por ejemplo «,»).

También hay en la ROM adicional algunas rutinas de exploración que pueden llamarse directamente.

PARAMS # 0701:	la rutina que comprueba los parámetros de LOAD, SAVE, etc. Todos los pará-

metros se pasan a D__STR1, N__STR1, S__STR1, L__STR1 y las variables comprendidas entre HD__00 y HD__11 en tiempo de ejecución.

EVALBC # 061E: evalúa un número de 16 bits que se retorna en el registro BC y en D__STR1.

CHKEND # 05B7: comprueba si se llega al fin de una sentencia. Si no ocurre esto, se produce un error. Si se llega al final de la sentencia, se ejecuta una instrucción RET en tiempo de ejecución. Si no se está en tiempo de ejecución, la dirección de retorno se elimina y el control se devuelve a la ROM de 16 K a través de 05C1.

CHKDRV 066D: Comprueba si el contenido de D__STR1 es un número comprendido entre el 1-8. En caso de que el número no esté entre esos dos se produce el error Número de unidad no válido **(Invalid drive number)**

CHKST$ 062F: Evalúa una cadena. Si se está en tiempo de ejecución, se comprueba si su longitud es inferior a 11 caracteres y se almacenan los parámetros en las posiciones que van de N__STR1 y T__STR1.

Una vez se ha comprobado toda la sintaxis y ha resultado correcta, se realiza un salto a 05C1H (a menos que se realice un retorno indirecto a través de CHKEND). Cuando se ha ejecutado un comando sin problemas, se debe realizar un salto similar. Se borra la pila y el código de error y se retorna a la ROM DE 16 K.

Como se puede ver, añadir nuevos comandos es una tarea difícil pero que vale la pena. Abre las puertas del Spectrum a todo tipo de programas de utilidad y ofrece la posibilidad de emplear el editor de líneas del BASIC para entrar líneas para otros lenguajes o ensambladores.

Lo que viene a continuación es el listado correspondiente a los comandos adicionales introducidos en la primera parte del capítulo.

Este listado no es independiente de la posición de carga ni se coloca de nuevo por sí mismo: el cargador del BASIC altera todos los bytes de las instrucciones «CALL» a medida que los va cargando con instrucciones POKE.

```
0010;                VECTOR PARA CAMBIAR SINTAXIS
0030          ORG    40000         ORIGEN DE DEPURACION
0040          CP     5CH           COMPROBAR PARA "*"-206
0050 ERROR    JP     NZ,01F0H      ERROR SI NO ES "*"
0060          RST    10H
0070          DEFW   0020H         LLAMAR A NEXT-CHAR
0080          OR     20H           CONVERTIRLO EN MINUSCULAS
0090          CP     "s"
0100          JR     Z,SAVE        SALTAR SI "s" O "S"
0110          CP     "l"
0120          JR     Z,LOAD        SALTAR SI "l" O "L"
0130          CP     "v"
0140          JR     Z,VERIF       SALTAR SI "v" O "V"
0150          CP     "m"
0160          JR     Z,MERGE       SALTAR SI "m" O "M"
0170          CP     "c"
0180          JR     Z,CAT         SALTAR SI "c" O "C"
0190          CP     "r"
0200          JR     Z,RUN         SALTAR SI "r" O "R"
0210          JR     ERROR         EN CASO CONTRARIO REALIZAR
                                   UN ERROR
0220 SAVE     SET    5,(IY+124)    INDICAR SAVE
0230          CALL   VARS          EVALUAR LA CADENA
0240          JP     0836H         REALIZAR EL SAVE
0250 LOAD     SET    4,(IY+124)    INDICAR LOAD
0260 DOIT     CALL   VARS          EVALUAR LA CADENA
0270          JP     08A5H         REALIZAR LOAD/VERIFY/MERGE
0280 VERIF    SET    7,(IY+124)    INDICAR VERIFY
0290          JR     DOIT          REALIZARLO
0300 MERGE    SET    6,(IY+124)    INDICAR MERGE
0310          JR     DOIT          REALIZARLO
0320 CAT      LD     HL,5CD6H      HL=D_STR1
0330          LD     (HL),1        ASIGNARLE LA UNIDAD 1
0340          INC    HL
0350          LD     (HL),0        PONER A CERO EL BYTE ALTO
                                   DE D_STR1
```

```
0360           INC  HL
0370           LD   (HL),2      ASIGNARLE LA CORRIENTE 2
0380           RST  10H
0390           DEFW 0020H       LLAMAR A NEXT-CHAR
0400           CP   13
0410           JR   Z,CAT2      SALTAR SI NEWLINE
0420           CP   ":"
0430           JR   Z,CAT2      SALTAR SI SE TRATA DE UN
                                SEPARADOR
0440           CALL 061EH       EVALUAR UN NUMERO
0450 CAT2      JP   04A9H       REALIZAR UN CAT
0460 RUN       RST  10H
0470           DEFW 0020H       LLAMAR A NEXT-CHAR
0480           OR   20H         CONVERTIR EN MINUSCULA
0490           CP   "u"
0500 ERR2      JP   NZ,0028H    ERROR SI NO ES "u"
0510           RST  10H
0520           DEFW 0020H       LLAMAR A NEXT-CHAR
0530           OR   20H
0540           CP   "n"
0550           JR   NZ,ERR2     ERROR SI NO ES "n"
0560           RST  10H
0570           DEFW 0020H       LLAMAR A NEXT-CHAR
0580           CALL 05B7H       LLAMAR.A CHKEND
0590           JP   0A95H       CARGAR EL PROGRAMA "RUN"
0600 VARS      LD   A,"M"
0610           LD   (5CD9H),A   L_STR1="M"
0620           RST  10H
0630           DEFW 0020H       LLAMAR A NEXT_CHAR
0640           RST  10H
0650           DEFW 1C8CH       LLAMAR CLASS_0A(CADENA)
0660           RST  10H
0670           DEFW 0018H       LLAMAR A GET_CHAR
0680           RST  18H         COMPROBAR LA SINTAXIS
0690           JP   Z,0723H     SALTAR LO SIGUIENTE SI
                                SOLO SE COMPRUEBA LA
                                SINTAXIS
0700           RST  10H
0710           DEFW 2BF1H       LLAMAR A STK_FETCH(BC=
                                LONGITUD,
0720           LD   HL,11       DE=INICIO)
```

```
0730        AND  A
0740        SBC  HL,BC
0750        JP   C,064CH     ERROR SI ES MAYOR DE 11
                             CARACTERES
0760        LD   A,B
0770        OR   C
0780        JP   Z,064CH     ERROR SI ES UNA CADENA
                             NULA
0790        LD   A,(DE)      A=CODIGO DEL PRIMER
                               CARACTER
0800        SUB  "0"         A=NUMERO DE LA UNIDAD
0810        CP   1
0820        JR   C,INVDR     ERROR SI <1
0830        CP   9
0840        JR   NC,INVDR    ERROR SI >8
0850        LD   (5CD6H),A   ALMACENAR EL NUMERO EN
                             D_STR1
0860        XOR  A
0870        LD   (5CD7H),A   BORRAR EL BYTE ALTO DE
                             D_STR1
0880        DEC  BC          DECREMENTAR LA LONGITUD
0890        INC  DE          INCREMENTAR LA POSICION
                             INICIAL
0900        LD   (5CDAH),    ALMACENAR LA LONGITUD
                 BC
0910        LD   (5CDCH),    ALMACENAR EL INICIO
                 DE
0920        RST  10H
0930        DEFW 0018H       LLAMAR A GET_CHAR
0940        JP   0723H       COMPROBAR LOS ARGUMENTOS
                             Y RETORNAR (RET)
0950 INVDR  RST  20H         REALIZAR OTRO ERROR
0960        DEFB 4           BYTE PARA NUM. DE UNIDAD
                             NO VALIDO
```

Como puede comprobarse, he saltado a algunas posiciones extrañas de la ROM adicional. No hay que realizar llamadas a estas posiciones mediante el código especial 32H puesto que utilizan las rutinas de exploración de líneas y retornan al BASIC (y no al código máquina que las ha llamado).

APENDICES

Apéndice A

LAS VARIABLES DEL SISTEMA DEL INTERFACE

Los siguientes 58 bytes se insertan en el mapa de memoria junto a las variables normales del sistema, desarrollando las funciones que se detallan:

Decimal	Hex	Nombre	Utilización
23734	5CB6	FLAGS3	bit 0: activado mientras se realiza un comando extendido bit 1: activado cuando se hace CLEAR # bit 2: activado al cambiar ERRSP bit 3: activado al utilizar la red bit 4: activado al hacer LOAD bit 5: activado al hacer SAVE bit 6: activado al hacer MERGE bit 7: activado al hacer VERIFY
23735	5CB7	VECTOR	Utilizado para ampliar el intérprete. Normalmente 01F0.
23737	5CB9	SBRT	Rutina utilizada por la ROM adicional para llamar a las rutinas de la ROM normal que tengan la forma: LD, HL, valor CALL rutina LD(5CBAH), HL RET
23747	5CC3	BAUD	velocidad del RS232 en baudios. Inicialmente 000CH (aproximadamente 19200) (3500000/26* [vel. en baudios]) — 2

23749	5CC5	NTSTAT	Número de estación de la red (1-64)
23750	5CC6	IOBORD	Color del borde de la pantalla durante operaciones de Entrada/Salida
23751	5CC7	SER__FL	Area de trabajo del RS232: el primer byte es un señalizador, el segundo el carácter introducido.
23753	5CC9	SECTOR	Area de trabajo del Microdrive. Se utiliza normalmente para contar sectores, empezando por FFH ó 04FBH.
23755	5CCB	CHADD__	Almacenamiento temporal de CH__ADD mientras se realiza una comprobación de la sintaxis extendida de líneas.
23757	5CCD	NTRESP	Código de respuesta de la red —número de la estación receptora— el reconocimiento inicial del encabezamiento de 8 bytes de la red.
23758	5CCE	NTDEST	Número de la estación de la red que es la destinataria.
23759	5CCF	NTSRCE	Número de la estación de la red emisora
23760	5CD0	NTNUMB	Número del bloque de la red (0-65535)
23762	5CD2	NTTYPE	Código de tipo del encabezamiento de la red (0: datos, 1: EOF)
23763	5CD3	NYLEN	Longitud del bloque de datos de la red (0-255)
23764	5CD4	NTDCS	Suma de control («Checksum») del bloque de datos de la red
23765	5CD5	NTHCS	Suma de control del encabezamiento de la red (de NTDEST a NTDCS)
23766	5CD6	D__STR1	Inicio del identificador de fichero (8 bytes), número de unidad (2 bytes), (1-8), número del destinatario o velocidad en baudios
23768	5CD8	S__STR1	Número de corriente (0-15)
23769	5CD9	L__STR1	Identificador del canal (mayúsculas)
23770	5CDA	N__STR1	Longitud del nombre del fichero
23772	5CDC	T__STR1	Inicio del nombre del fichero (normalmente en el área de trabajo)

23774	5CDE	D__STR2	Inicio del segundo identificador de fichero (8 bytes)
23782	5CE6	HD__00	Tipo del fichero: 0. programa 1. conjunto numérico 2. conjunto de cadenas 3. bytes
23783	5CE7	HD__0B	Longitud de los datos
23785	5CE9	HD__0D	Inicio de los datos
23787	5CEB	HD__0F	Longitud del programa (o nombre del conjunto)
23789	5CED	HD__11	Número de línea para autoiniciación puesta en marcha (también empleado por el código especial 32H)
23791	5CEF	COPIES	Número de copias múltiples realizadas por SAVE. Toma el valor 1 después de un SAVE.

Notas:

FLAGS3 toma normalmente el valor cero cuando la ROM adicional sale, debido a la paginación, del espacio de memoria. Puede direccionarse mediante IY + 124.

Los parámetros de la rutina SBRT se pasan a la ROM adicional mediante POKEs.

La instrucción de retorno (RET) va siempre a la posición 8 pero siempre hay un 0000 en la posición siguiente de la pila para distinguir cuándo se trata de un error.

Las variables de nombre HD__00 a HD__11 se extraen de las variables equivalentes utilizadas por las rutinas de cassette de la ROM original y se direccionan mediante IX + 0 a IX + 11H.

Apéndice B

LISTADOS EN ENSAMBLADOR

En este Apéndice se encuentran los listados en Ensamblador del Z80 de las rutinas utilizadas a lo largo del libro. Están basadas en originales creados mediante el ensamblador «Picturesque Editas» que indica los números en hexadecimal poniendo tras ellos una «H». Los números de línea los utiliza el Ensamblador y no tienen aquí significado algunc. Se han incluido comentarios en la parte derecha del listado. Todos los «ORG» son arbitrarios puesto que la mayoría de las rutinas son independientes de la posición o se sitúan de nuevo a sí mismas. En el caso de que una rutina sea auto-reubicable, el registro BC contiene su dirección inicial.

Corriente 14-z$

Con esta rutina se crea un nuevo canal «Z» y se asigna a la corriente 14 que se supone está cerrada. También hay que suponer que ya se han creado las 58 variables adicionales del sistema. Por esta razón, se incluye el comando CLOSE # en el cargador del BASIC pues se encarga de realizar todas estas operaciones.

```
;                   "PRINT RUTINA DE CADENA"
;                   "RUTINA PARA HACER QUE"
;                   "#14 INSERTE CARACTERES
                     EN Z$"
;                   "(C) A.PENNELL 1983"
PROG    EQU         5C53H
VARS    EQU         5C4BH
```

```
0080            ORG         40000
0090    INICIOLD            HL,(PROG)
0100    DEC     HL          HL=PROG-1=FIN DE CHANS
0110    PUSH    BC          ALMACENAR (SAVE) LA DIREC-
                            CION DE INICIO EN LA PILA
0120    PUSH    HL          ALMACENAR (SAVE) EN LA PI-
                            LA EL INICIO DEL AREA OCU-
                            PADA POR LOS NUEVOS DATOS
0130    LD      BC,11       SON NECESARIOS 11 BYTES
0140    CALL    1655H       LLAMAR A MAKE-ROOM
0150    POP     DE          DE=NUEVO PUNTO DE INICIO
                            DE DATOS
0160    LD      HL,CARSAL-INICIO
0170    POP     BC          SACAR INICIO DE LA PILA
0180    ADD     HL,BC       HL=CARSAL
0200    PUSH    DE          ALMACENAR (SAVE) EL
                            INICIO DE LOS DATOS
0210    EX      DE,HL
0220    LD      (HL),E      ALMACENAR CARSAL EN LA
                            NUEVA AREA
0230    INC     HL
0240    LD      (HL),D
0250    INC     HL
0260    EX      DE,HL
0270    LD      BC,ETIQUETA-CARSAL
0280    ADD     HL,BC       HL=ETIQUETA
0290    LD      BC,9        NUEVE BYTES MAS DE DATOS
0300    LDIR                COPIAR LOS DATOS EN CHANS
0310    POP     HL          HL=INICIO DE DATOS
0320    INC     HL
0330    LD      BC,(5C4FH)  BC=CHANS
0340    AND     A
0350    SBC     HL,BC
0360    LD      (5C32H),HL  ALMACENAR EL DESPLAZA-
                            MIENTO DENTRO DE STRMS
                            QUE LE CORRESPONDE A
                            LA CORRIENTE #14
0370    LD      BC,0
0380    RET                 VOLVER AL BASIC CON 0
                            EL RESTO DE DATOS VA A
                            IR A CHANS:
```

```
0390 ETIQ    DEFW  15C4H        LA RUTINA DE 'ENTRADA'
0400         DEFM  "Z"          EL NOMBRE DEL CANAL
0410         DEFW  28H          SALIDA DE LA ROM
                                ADICIONAL
0420         DEFW  28H          ENTRADA DE LA ROM
                                ADICIONAL
0430         DEFW  11           EL NUMERO DE BYTES
                                LA RUTINA DE SALIDA:
0440 CARSAL  PUSH  AF           ALMACENAR (SAVE) EL
                                CODIGO DE CARACTER
0450         LD    HL,(VARS)
0460 L1      LD    A,(HL)
0470         CP    5AH
0480         JR    Z,ENCON-     SALTAR SI SE HA
                     TRADO      ENCONTRADO Z$
0490         CP    80H
0500         JP    Z,0670H      'VAR NO ENCONTRADA' SI
                                NO HAY MAS
0510         CALL  19B8H        LLAMAR A NEXT-ONE
0520         EX    DE,HL        HL=INICIO DE LA PROXIMA
                                VARIABLE
0530         JR    L1           VOLVER AL INICIO DEL
                                BUCLE
0540 ENCON-  INC   HL           HL=BYTE BAJO DE LA
     TRADO                      LONGITUD DE LA CADENA
0550         LD    C,(HL)
0560         INC   HL           HL=BYTE ALTO DE LA
                                LONGITUD DE LA CADENA
0570         LD    B,(HL)       BC=LONGITUD DE LA CADENA
0580         INC   BC           INCREMENTAR LONGITUD
0590         PUSH  BC           ALMACENAR (SAVE) LA NUEVA
                                LONGITUD
0600         PUSH  HL           ALMACENAR (SAVE) LA
                                POSICION DEL PRIMER
                                CARACTER
0610         ADD   HL,BC        HL=FINAL DE LA CADENA
0620         CALL  1652H        LLAMAR A ONE_SPACE(HACER
                                SITIO PARA CARACTER)
0630         INC   HL           HL=NUEVO ESPACIO
0640         EX    DE,HL        DE=NUEVO ESPACIO
```

```
0650        POP   HL          HL=POSICION DEL PRIMER
                              CARACTER
0660        POP   BC          BC=NUEVA LONGITUD
0670        LD    (HL),B      ALMACENAR LA NUEVA
                              LONGITUD
0680        DEC   HL
0690        LD    (HL),C
0700        POP   AF          SACAR DE LA PILA EL
                              CODIGO DEL CARACTER
0710        LD    (DE),A      ALMACENAR EL CARACTER
                              EN LA CADENA
0720        AND   A           PONER EL ACARREO A CERO
                              (PARA EVITAR ERRORES)
0730        RET               VUELTA AL SISTEMA
                              OPERATIVO
0740        END
```

On EOF GOTO

Esta rutina cambia ERRSP para que apunte a una rutina conveniente de tratamiento de error.

```
0010;                "FIN DE FICHERO (ON EOF GOTO)"
0020;                "CODIGO INDEPENDIENTE DE LA POSICION
                      (PIC CODE)"
0040        ORG   40000
0050 INIC   LD    HL,ORIGEN-INIC CAMBIAR ERRSP:
0060        ADD   HL,BC       HL=ORIGEN
0070        EX    DE,HL       DE=ORIGEN
0080        LD    HL,(ERRSP)  HL=ERRSP
0090        LD    (HL),E      ALMACENAR LA NUEVA RUTINA
                              DE ERROR
0100        INC   HL          EN LA POSICION CORRESPON-
0110        LD    (HL),D      DIENTE DENTRO DE LA PILA
0120        RST   8           ASEGURARSE DE QUE LAS
0130        DEFB  31H         VARIABLES DEL INTERFACE
                              ESTAN AHI
0140        LD    BC,0
0150        RET               RETORNO AL BASIC CON 0 RU-
                              TINA DE GESTION DE ERRORES:
```

```
0160 ORIGEN LD    HL,(ERRSP) HL=POSICION DE LA PILA
0170        LD    A,(5C3AH)  A=CODIGO DE ERROR
0180        CP    EOF        ES UN EOF?
0190        JP    NZ,1303H   SALTAR A LA RUTINA DE
                             ERROR DE LA ROM SI NO
                             LO ES
0200        LD    E,(HL)     EN OTRO CASO, DE=ORIGEN
0210        INC   HL
0220        LD    D,(HL)
0230        PUSH  DE         PONER ORIGEN EN EL FONDO
                             DE LA PILA
0240        CALL  16B0H      BORRAR LAS AREAS DE
                             TRABAJO Y EDICION
0250        RES   5,(IY=37H) INDICAR QUE SE ESTA LISTO
                             PARA OTRA TECLA
0260        CALL  0D6EH      BORRAR LA PARTE INFERIOR
                             DE LA PANTALLA Y ABRIR
                             LA CORRIENTE 0
0270        LD    HL,(5C45H) HL=NUMERO DE LINEA EN UN
                             MOMENTO DADO
0280        LD    (5CC9H),HL ALMACENARLO EN SECTOR
0290        LD    DE,LINEA   DE=LINEA A LA QUE HAY QUE
                             SALTAR
0300        LD    HL,5C42H   HL=NEWPPC
0310        LD    (HL),E     ALMACENAR EL NUEVO NUMERO
                             DE LINEA
0320        INC   HL
0330        LD    (HL),D
0340        INC   HL
0350        LD    (HL),1     HACER NSPPC=1, ES DECIR,
0370        JP    1B7DH      QUE LA PRIMERA INSTRUC-
                             CION SALTE A LA ROM
0380 ERRSP  EQU   5C3DH
0390 EOF    EQU   7          CODIGO QUE INFORME DE EOF
0400 LINEA  EQU   1000       NUMERO DE LINEA DEL ERROR
0410        END
```

OPEN # cualquier cosa

Esta rutina hace que se abra para un canal del Microdrive la corriente que se especifique. Básicamente se trata de la rutina del co-

mando OPEN # a la que se han modificado las comprobaciones de error.

```
0010 ;              "OPEN CUALQUIER COSA"
0030 ;
0040 ;
0050 OPENM   EQU    22H          CODIGOS ESPECIALES
0060 CLOSM   EQU    23H
0070 MOTOR   EQU    21H
0080         ORG    40000
0090 ORIGEN  LD     A,(5CD8H)    A=S_STR1=NUMERO DE
                                 CORRIENTE
0100         CALL   1727H        LLAMAR A STR_DATA1
0110         LD     HL,17
0120         XOR    A
0130         SBC    HL,BC
0140         LD     BC,0
0150         RET    C            VUELTA AL BASIC CON 0 SI
                                 LA CORRIENTE YA ESTABA
                                 ABIERTA
0160         LD     (5CD7H),A    PONER A CERO EL BYTE ALTO
                                 DE D_STR1
0170         LD     HL,10
0180         LD     (5CDAH),HL   HACER QUE LA LONGITUD DEL
                                 NOMBRE DEL FICHERO SEA=10
0190         LD     HL,(5C7BH)
0200         LD     (5CDCH),HL   HACER QUE INICIO DEL
                                 NOMBRE = AREA DE GRAFICOS
                                 DEL USUARIO
0210         LD     A,(5CD8H)    A=NUMERO DE CORRIENTE
0220         ADD    A            CALCULAR EL STRM
                                 CORRESPONDIENTE
0230         LD     HL,5C16H
0240         LD     E,A
0250         LD     D,0
0260         ADD    HL,DE        HL=POSICION DE LA
                                 CORRIENTE
0270         EXX
0280         PUSH   HL           ALMACENAR (SAVE) H'L'
0290         EXX
0300         PUSH   HL           SALVAR LA POSICION STRM
```

```
0310        RST   8            ABRIR UN CANAL "M"
                               TEMPORAL
0320        DEFB  OPENM
0330        BIT   0,(IX+24)
0340        JR    Z,LECT       SALTAR SI ES UN FICHERO
                               DE ALTURA
0350        XOR   A            FICHERO NO ENCONTRADO
0360        RST   8            ASI QUE DESCONECTAR TODOS
                               LOS MOTORES
0370        DEFB  MOTOR
0380        RST   8            Y LIBERAR EL AREA
0390        DEFB  2CH
0400        POP   HL           RESTAURAR LA PILA
0410        EXX
0420        POP   HL           SACAR DE LA PILA H'L'
0430        EXX
0440        LD    BC,1
0450        RET                VOLVER CON 1 SI NO SE
                               HA ENCONTRADO
0460 LECT   RES   7,(IX+4)     HACERLO PERMANENTE
0470        XOR   A
0480        PUSH  HL           ALMACENAR (SAVE) EL
                               DESPLAZAMIENTO DE LA
                               CORRIENTE
0490        RST   8            DESCONECTAR TODOS LOS
                               MOTORES
0500        DEFB  MOTOR
0510        POP   DE           DE=DESPLAZAMIENTO
0520        POP   HL           HL=POSICION
0530        LD    (HL),E       ALMACENAR EL NUEVO
                               DESPLAZAMIENTO
0540        INC   HL           EN EL AREA STRMS
0550        LD    (HL),D
0560        LD    A,(IX+67)    A=RECFLG
0570        AND   4            ENMASCARAR EL BIT QUE
                               INDICA FICHERO DE TIPO
                               PRINT
0580        ADD   2            SUMAR 2
0590        LD    C,A          DEJAR EN BC
0600        LD    B,0
0610        EXX
```

```
0620          POP   HL          EXTRAER H'L' DE LA PILA
0630          EXX
0640          RET               VOLVER CON UN 2 SI SE
                                TRATABA DE UN FICHERO DE
                                TIPO PRINT O CON UN 6 EN
                                CASO CONTRARIO
0650          END
```

Rutina de estado

Esta rutina es muy similar a la que está en la ROM adicional con el nombre de MOTOR (posiciones X182AH a X1871H) pero con la gestión de errores modificada. Su misión es determinar si está presente un determinado Microdrive o no. En caso de que lo esté, se determina si el cartucho está protegido contra escritura. Las etiquetas utilizadas se basan en las posiciones equivalentes de la ROM adicional. La posición ORG es una de origen de depuración —**no** se trata, pues, de una rutina cuyo código sea independiente de la posición—. El cargador del BASIC se encarga de realizar la reubicación de los cuatro CALLs.

(No puedo asegurar qué hace exactamente cada instrucción de Entrada/Salida de los «ports»: las notas indicadas aquí son aproximadas).

```
0010;                RUTINA DE ESTADO
0020 MOTOR    EQU    21H          CODIGO ESPECIAL DEL MOTOR
0030          ORG    40000        ORIGEN DE DEPURACION
0040          LD     A,(5CD6H)    A=D-STR1=NUMERO DE UNIDAD
0050          DI                  NO SE PERMITEN LAS INTE-
                                  RRUPCIONES
0060          JR     X182A        SALTAR ESTO
0070 X1806    LD     HL,1388H     HL=5000
0080 X1809    DEC    HL
0090          LD     A,L
0100          OR     H
0110          JR     NZ,X1809     ESPERAR UN POCO
0120          LD     HL,1388H
0130 X1811    LD     B,6
0140 X1813    IN     A,(0EFH)     LEER DEL "PORT" EF
```

```
0150             AND    4            SOLO EL BIT 2
0160             JR     NZ,X1820     SALTAR SI LA UNIDAD NO
                                     ESTA PRESENTE
0170             DJNZ   X1813        COMPROBARLO SEIS VECES
0180             JR     PRESN        LO ESTA, POR LO TANTO,
                                     RETORNAR
0190 X1820       DEC    HL           DECREMENTAR EL CONTADOR
0200             LD     A,H
0210             OR     L
0220             JR     NZ,X1811     REPETIRLO 5000 VECES
0230             LD     BC,0         NO SE PUEDE CONECTAR POR
0240             JR     NOTPR        TANTO, VOLVER CON 0
0250 X182A       LD     DE,0100H     D=1,E=0
0260             NEG                 NEGAR EL NUMERO DE LA
                                     UNIDAD
0270             ADD    9            A=9-NUMERO DE LA UNIDAD DE
                                      MICRODRIVE
0280             LD     C,A          C=UNIDAD SELECCIONADA
0290             LD     B,8          B=CONTADOR DE LA UNIDAD
0300 X1835       DEC    C            DECREMENTAR LA UNIDAD
                                     SELECCIONADA
0310             JR     NZ,X184B     SALTAR SI NO SE TRATA DEL
                                     QUE ESTAMOS INVESTIGANDO.
                                     PARA PONER EN MARCHA
                                     MOTOR:
0320             LD     A,D
0330             LD     (0F7H),A     ENVIAR UN 1 AL "PORT" F7
                                     -CONECTAR-
0340             LD     A,0EEH
0350             OUT    (0EFH),A     ENVIAR EE AL "PORT" EF
0360             CALL   X1867        ESPERAR UN POCO
0370             LD     A,0ECH
0380             OUT    (0EFH),A     ENVIAR EC AL "PORT" EF
0390             CALL   X1867        ESPERAR UN POCO
0400             JR     X185C        HAGASE LO SIGUIENTE,
                                     DESCONECTAR UNA UNIDAD
0410 X184B       LD     A,0EFH
0420             OUT    (0EFH),A     ENVIAR EF AL "PORT" EF
0430             LD     A,E
0440             OUT    (0F7H),A     ENVIAR UN 0 AL "PORT" F7
                                     -DESCONECTAR-
0450             CALL   X1867        ESPERAR UN POCO
```

```
0460          LD     A,0EDH
0470          OUT    (0EFH),A     ENVIAR ED AL "PORT" EF
0480          CALL   X1867        ESPERAR UN POCO
0490 X185C    DJNZ   X1835        HACER LO MISMO CON LAS
                                  8 UNIDADES
0500          LD     A,D
0510          OUT    (0F7H),A     ENVIAR UN 1 AL "PORT" F7
0520          LD     A,0EEH
0530          OUT    (0EFH),A     ENVIAR EE AL "PORT" EF
0540          JR     X1806        COMPROBAR SU ESTADO
0550 X1867    PUSH   BC           RUTINA DE RETARDO:
0560          PUSH   AF           ALMACENAR (SAVE)
                                  REGISTROS
0570          LD     BC,0087H     BC=135
0580 X18FB    DEC    BC
0580 X18FB    DEC    BC
0590          LD     A,B
0600          OR     C
0610          JR     NZ,X18FB     REALIZAR EL BUCLE 135
                                  VECES
0620          POP    AF           RESTAURAR REGISTROS
0630          POP    BC
0640          RET                 VOLVER
0650 PRESN    IN     A,(0EFH)     LA UNIDAD ESTA PRESENTE,
0660          AND    1            ASI QUE, COMPROBAR LA
                                  LENGUETA
0670          LD     BC,1
0680          JR     NZ,NOTPR     SALTAR CON UN 1 SI LA
                                  LENGUETA ESTA ALLI.
0690          INC    BC           EN OTRO CASO 2
0700 NOTPR    PUSH   BC           SALVAR EL VALOR DE RETORNO
0710          XOR    A
0720          RST    8
0730          DEFB   MOTOR        DESCONECTAR TODOS LOS
                                  MOTORES
0740          POP    BC           RESTAURAR EL VALOR
0750          RET                 VUELTA AL BASIC
0760          END
```

On error goto

Esta rutina cambia ERR__SP para que apunte a una nueva rutina de gestión de errores. Esto no basta para los errores del Interface: para ellos hay que activar el bit 2 de FLAGS3 pues, de otra forma, se emplearía la rutina normal de gestión de errores. Si el error tiene lugar cuando, por paginación, la ROM ocupa el espacio de memoria, HL valdrá 81H.

```
0010 ;          ON ERR GOTO"
0020 ;          VERSION PARA INTERFACE
0030 ERRSP  EQU   5C3DH       INICIALIZAR LAS
                              CONSTANTES
0040 SECTR  EQU   5CC9H
0050 LINEA  EQU   1000        SALTO DE LA LINEA DE
                              ERROR
0060        ORG   40000       ORIGEN ARBITRARIO
0070 INICIO LD    HL,ORIGEN-INICIO
0080        ADD   HL,BC       HL=INICIO
0090        EX    DE,HL       DE=INICIO
0100        LD    HL,(ERRSP)  HL=POSICION DE ERROR EN
                              LA PILA
0110        LD    (HL),E      ALMACENAR LA DIRECCION DE
0120        INC   HL          LA NUEVA RUTINA DE GES-
                              TION DE ERROR EN LA PILA
0130        LD    (HL),D
0140        RST   8
0150        DEFB  31H         CREAR LAS VARIABLES
                              ADICIONALES
0160        LD    BC,0
0170        RET               VOLVER CON 0
                              LA NUEVA RUTINA DE
                              GESTION DE ERROR:
0180 ORIGEN LD    (SECTR),HL  ALMACENAR EL VALOR DE HL
0190        LD    HL,(ERRSP)
0200        LD    DE,1303H    DE=RUTINA DE ERROR DE LA
                              ROM ANTIGUA
0210        PUSH  DE          GUARDARLA EN LA PILA PARA
                              IMPEDIR QUE LA PROXIMA
                              VEZ SE ACTIVE ESTA
                              FUNCION
```

```
0220    CALL  16B0H        BORRAR LAS DISTINTAS
                           AREAS
0230    RES   5,(IY+37H)   BORRAR EL BIT DE LA TECLA
                           EN FLAGS
0240    CALL  0D6EH        BORRAR LA PARTE INFERIOR
                           DE LA PANTALLA Y ABRIR
                           LA CORRIENTE 0
0250    LD    HL,17B9H     HL=RUTINA DE LA ROM ADI-
                           CIONAL QUE RECUPERA TODOS
                           LOS CANALES TEMPORALES Y
                           DESCONECTA TODOS LOS
                           MOTORES
0260    LD    (5CEDH),HL   ALMACENAR EN HD-11
0270    LD    A,(5C3AH)    A=NUMERO DE ERROR
0280    LD    (23763),A    ALMACENARLO EN NTLEN
0290    PUSH  AF           ALMACENAR (SAVE) EL
                           CODIGO DE ERROR
0300    RST   8
0310    DEFB  32H          LLAMAR X17B9H
0320    LD    HL,0081H
0330    LD    DE,(SECTR)   DE=VALOR DE HL DESPUES
                           DEL ERROR
0340    POP   AF           A=CODIGO DE ERROR DE LA
                           ROM ANTIGUA
0350    AND   A            PONER A CERO EL BIT DE
                           ACARREO
0360    SBC   HL,DE
0370    JR    NZ,OLD       SALTAR SI EL HL<>81H (ES
                           DECIR, SI SE TRATA UN
                           ERROR DE LA ROM ANTIGUA)
0380    BIT   0,(IY+124)   COMPROBAR FLAGS3
0390    JR    NZ,OLD       SALTAR A UN ERROR DE LA
                           ROM ANTIGUA SI ESTA EN
                           MEDIO DE UNA INSTRUCCION
0400    CP    7
0410    JR    Z,OLD        SALTAR SI ERA UN EOF
0420    LD    A,100        SE TRATA DE UN ERROR DEL
                           INTERFACE, POR TANTO,
0430    LD    (23763),A    ALMACENAR 100 EN NTLEN
0440    LD    A,"PRINT"
0450    RST   10H          IMPRIMIR UN "PRINT"
```

```
0460             JR     PRNTN       IMPRIMIR NUMEROS DE LINEA
                                    ETC. ERROR DE LA ROM
                                    ANTIGUA:
0470             INC    A
0480             LD     B,A         ALMACENAR EN B EL NUMERO
                                    DE ERROR
0490             CP     10
0500             JR     C,2
0510             ADD    7           FORMAR EL CARACTER DEL
                                    MENSAJE DE ERROR
0520             CALL   15EFH       IMPRIMIRLO
0530             LD     A," "
0540             RST    10H         IMPRIMIR UN ESPACIO
0550             LD     A,B
0560             LD     DE,1391H
0570             CALL   0C0AH       IMPRIMIR EL MENSAJE DE
                                    ERROR
0580 PRNTN       XOR    A
0590             LD     DE,1536H
0600             CALL   0C0AH       IMPRIMIR ","
0610             LD     BC,(5C45H)  BC=PPC=LINEA QUE SE ESTA
                                    TRATANDO
0620             LD     (SECTR),BC  ALMACENAR EN SECTOR
0630             CALL   1A1BH       IMPRIMIR EL NUMERO DE
                                    LINEA
0640             LD     A,":"
0650             RST    10H         IMPRIMIR DOS PUNTOS (":")
0660             LD     C,(IY+13)
0670             LD     B,0
0680             CALL   1A1BH       IMPRIMIR EL NUMERO DE
                                    LA INSTRUCCION
0690             LD     HL,5C3BH    HL=FLAGS
0700             RES    5,(HL)
0710             EI                 PERMITIR INTERRUPCIONES
0720 ESPERA      BIT    5,(HL)
0730             JR     Z,WAIT      ESPERAR A QUE SE PULSE
                                    UNA TECLA
0740             LD     HL,5C42H    HL=NEWPPC
0750             LD     DE,LINE     DE=LINEA DE ERROR A LA
                                    QUE HAY QUE SALTAR
0760             LD     (HL),E      ALMACENAR LA LINEA
```

```
0770          INC   HL
0780          LD    (HL),D
0790          INC   HL
0800          LD    (HL),1      CONVERTIR EN LA
                                INSTRUCCION 1
0810          LD    (IY+0),255  BORRAR EL ERROR
0820          LD    (IY+124),0  BORRAR (PONER A CERO)
                                FLAGS3
0830          CALL  0D6EH       BORRAR LA PARTE INFERIOR
                                DE LA PANTALLA
0840          JP    1B7DH       SALTAR A LA ROM DE 16K
0850          END
```

Rutina TAB para el RS232

Esta rutina cambia los datos del canal «P» para que LPRINTs, etc. se puedan enviar a la vía de acceso (port) del RS232 de forma similar al canal «T» pero con la función TAB. Se utilizan tres variables propias del sistema: WIDTH = anchura de la impresora, POSN = posición de la cabeza de impresión en un momento dado y CONTR = trabaja como señalizador. Cuando se interpreta la función TAB, se envía primero el carácter 23 y luego el bit menos significativo (LSB) del siguiente número seguido del bit más significativo (MSB).

```
0010;         "RUTINA DE TAB PARA RS232"
0020          ORG   23296       SITUARLA EN LA MEMORIA
                                INTERMEDIA (BUFFER) DE
                                LA IMPRESORA
0030 INIC     LD    HL,(23631)  HL=CHANS
0040          LD    BC,15
0050          ADD   HL,BC       HL=AREA DEL CANAL "P"
0060          LD    DE,ORIGEN   DE=NUEVA RUTINA DE SALIDA
0070          LD    (HL),E      ALMACENAR LA NUEVA RUTINA
0080          INC   HL          POSICION EN EL AREA "P"
0090          LD    (HL),D
0100          LD    BC,0
0110          LD    HL,POSN
0120          LD    (HL),B      DAR A POSN EL VALOR 0
```

```
0130         INC   HL
0140         LD    (HL),B      DAR A CONTR EL VALOR 0
0150         RET               RETORNO CON 0
                               LA RUTINA DE SALIDA:
                               A=CODIGO
0160 ORIGEN  CP    " "
0170         JR    NC,NORM     SALTAR SI >=" "
0180         CP    13
0190         JR    NZ,NNL      SALTAR SI NO SE TRATA DE
                               NEWLINE
0200         LD    HL,CONTR    NEWLINE:
0210         BIT   0,(HL)      COMPROBAR EL SENALIZADOR
                               DE AVANCE DE LINEA
                               AUTOMATICO
0220         RES   0,(HL)      PONERLO A 0
0230         RET   NZ          EN CASO DE QUE ESTE
                               ACTIVADO, NO HACER
                               NEWLINE
0240 NEWLI   LD    HL,CONTR
0250         RES   0,(HL)      BORRAR EL SENALIZADOR
                               DE AVANCE DE LINEA
                               AUTOMATICO
0260         DEC   HL          HL=POSN
0270         LD    (HL),0      PONER A 0 POSN
0280         LD    A,13
0290         CALL  CARSAL      ENVIAR UN "NEWLINE" (13)
0300         LD    A,10
0310         JP    CARSAL      ENVIAR UN AVANCE DE LINEA
                               (LINE-FEED) (10)
0320 NNL     CP    23          ES EL CARACTER DE TAB?
0330         CCF
0340         RET   NZ          RETORNO SI NO LO ES
0350         LD    DE,TAB2     INSTRUCCION TAB
0360 BUCLE   LD    HL,(5C51H)  HL=CURCHL
0370         LD    (HL),E      ALMACENAR DE EN CURCHL,
0380         INC   HL          ES DECIR, CAMBIAR LA PO-
                               SICION DE SALIDA POR DE
0390         LD    (HL),D
0400         RET               RETORNO.
                               TRAER AQUI EL LSB DE LA
                               POSICION DE TAB:
```

```
0410 TAB2    LD     (5C0FH),A  ALMACENAR EL LSB EN
                               TVDATA2
0420         LD     DE,TAB3
0430         JR     REDO       EJECUTAR LA RUTINA DE SA-
                               LIDA TAB3. TRAER AQUI EL
                               MSB DE LA POSICION TAB:
0440 TAB3    LD     DE,ORIGEN
0450         CALL   REDO       VOLVER LA RUTINA DE
                               SALIDA A LA NORMAL
0460         LD     A,(5C0FH)  A=POSICION DE TAB
0470         LD     D,A        ALMACENAR EN D
0480 TAB4    LD     HL,WIDTH
0490         SUB    (HL)       A=TAB-WIDTH
0500         JP     NC,046CH   "FUERA DE RANGO" SI LA
                               ANCHURA NO ES SUFICIENTE
0510         INC    HL         HL=POSN
0520         LD     A,D        A=TAB
0530         SUB    (HL)       A=TAB-POSN
0540         PUSH   DE         ALMACENAR (SAVE) D
0550         CALL   C,NEWLI    HACER "NEWLINE" SI ES
                               NECESARIO
0560         POP    DE         RESTAURAR D
0570         LD     A,D        A=TAB
0580         SUB    (IY+POSY)  A=TAB-POSN
0590         RET    Z          RETORNAR SI ESTABA EN EL
                               LUGAR CORRECTO
0600         LD     B,A        B=NUMERO DE ESPACIOS
                               NECESARIOS
0610 TABB    LD     A," "
0620         PUSH   BC         ALMACENAR (SAVE) BC
0630         EXX               INTERCAMBIAR EL CONTENIDO
                               DE LOS REGISTROS
0640         RST    10H        IMPRIMIR UN ESPACIO
0650         EXX               VOLVER A INTERCAMBIAR EL
                               CONTENIDO DE LOS
                               REGISTROS
0660         POP    BC         RESTAURAR BC
0670         DJNZ   TABB       IMPRIMIR LOS ESPACIOS
                               APROXIMADOS
0680         RET               RETORNAR UNA VEZ HECHO
                               ESTO. TRAER AQUI LOS
                               CODIGOS "NORMALES"
```

```
0690 NORM    CP    0A5H
0700         JR    C,NORM2      SALTAR SI NO SE TRATA DE
                                UN COMANDO ESPECIAL
0710         SUB   0A5H
0720         JP    0C10H        QUITARLE EL CARACTER DE
                                ESPECIAL AL COMANDO
0730 NORM2   RES   0,(IY+CONY)  PONER A CERO EL INDICA-
                                DOR DE AVANCE DE LINEA
                                (LINE-FEED) AUTOMATICO
0740         LD    HL,5C3BH     HL=FLAGS
0750         RES   0,(HL)       PONER A CERO EL SENALIZA-
                                DOR DE "ESPACIO PRECEDEN-
                                TE" (LO QUE NO HACE LA
                                ROM)
0760         CP    " "
0770         JR    NZ,NSPAC     DEJARLO SI NO ES UN
                                ESPACIO, EN CASO CONTRA-
0780         SET   0,(HL)       RIO PONER EL SENALIZADOR
                                A UNO
0790 NSPAC   CP    128
0800         JR    C,CARSAL2    IMPRIMIR EL CARACTER SI
                                NO ES GRAFICO
0810         LD    A,"?"        SE TRATA DE UN CARACTER
                                GRAFICO, POR TANTO,
                                IMPRIMIR "?"
0820 CARSAL2 CALL  CARSAL       ENVIAR EL BYTE A LA
                                IMPRESORA
0830         LD    HL,POSN
0840         INC   (HL)         INCREMENTAR POSN
0850         LD    A,(HL)       A=NUEVA POSN
0860         DEC   HL           HL=WIDTH
0870         CP    (HL)         COMPARAR CON WIDTH
0880         RET   NZ           RETORNAR SI NO SON
                                IGUALES
0890         CALL  NEWLI        FIN DE LINEA, HAY QUE
                                HACER "NEWLINE" Y ACTIVAR
0900         SET   0,(IY+CONY)  EL SENALIZADOR DE AVANCE
                                DE LINEA (LINE-FEED)
                                AUTOMATICO
0910         RET                RETORNAR
                                ENVIAR UN CARACTER AL RS232
```

```
0920 CARSAL RST    8              CODIGO ESPECIAL
0930        DEFB   1EH            BYTE 2320P
0940        RET                   RETORNO
                                  CONSTANTES:
0950 WIDTH  EQU    23540          MEMORIA INTERMEDIA
                                  (BUFFER) DE LA IMPRESORA
0960 POSN   EQU    WIDTH+1
0970 CONTR  EQU    WIDTH+2
0980 POSY   EQU    0BBH           DESPLAZAMIENTOS DE IY
0990 CONY   EQU    0BCH
1000        END
```

Apéndice C

ERRORES DEL INTERFACE

Cuando el Interface está conectado, se utiliza una ROM adicional de 8 K para nuevas realizaciones. Hay que tener muy en cuenta que pueden surgir problemas al hacer uso de ella por causa de errores en el proceso de fabricación, especialmente en el anterior modelo de ROM.

1. Fallos en la comprobación de sintaxis

Cuando se preparó la ROM de 16 K original, la sintaxis para los comandos del Interface se anticipó incorrectamente. Desgraciadamente las formas sintácticas incorrectas —las de la ROM original— son aceptadas por el comprobador sintáctico pero no pueden ejecutarse.

Las formas sintácticas incorrectas son:

ERASE	<cadena>	p. ej.	ERASE «prueba»
MOVE	<cadena>, <cadena>	p. ej.	MOVE «a», «b»
FORMAT	<cadena>	p. ej.	FORMAT «prueba»
CAT	(sin ningún número)		

2. Comandos de color

Este no es un problema que se deba estrictamente a la ROM en el Interface sino a un fragmento de código en BASIC poco eficiente. Para solventarlo es necesario anteponer a cada instrucción de color una instrucción PRINT;

3. Inconsistencias en el BREAK

Parece existir en la antigua ROM una inconsistencia general en lo que se refiere a la forma en que hay que utilizar BREAK con las operaciones del Interface. Cuando se efectúan operaciones con el Microdrive o cuando se realiza una salida a través del RS232 hay que pulsar simultáneamente las teclas CAPS SHIFT y SPACE. En cambio, cuando se trabaja con la red o se recibe una entrada a través del RS232 basta con pulsar la tecla SPACE. En la ROM nueva siempre se deben pulsar las dos teclas al mismo tiempo.

4. Operaciones con el cassette

Cuando, durante la realización de una operación con el cassette, se hace servir un nombre de fichero que, por cualquier motivo, no sea correcto (por ejemplo, que tenga más de 10 caracteres) en vez de aparecer el mensaje **«Invalid filename»** (Nombre de fichero incorrecto) aparece el mensaje **«Nonsense in BASIC** (Sin sentido en BASIC) que no ayuda mucho a solventar el error producido.

5. Carga (LOAD) de un programa BASIC de gran tamaño

Si se intenta cargar un programa BASIC de gran tamaño y no hay espacio suficiente para él, se puede producir un error fatal para el sistema quedando, además, el motor en marcha. Desgraciadamente este error se mantiene en la nueva ROM.

Los siguientes problemas tienen que ver únicamente con la ROM antigua. Han sido corregidos y no se encuentran en la nueva.

6. Códigos especiales en lenguaje máquina

Dos de los códigos especiales en lenguaje máquina no funcionan. No se puede emplear READ__N porque afecta al señalizador de acarreo. Por otra parte, MAKE__M hace lo mismo que OPEN__M. Este error se debe a una entrada incorrecta en la tabla de saltos de los códigos especiales. Esta entrada se encuentra en la posición X19C9H.

7. Dobles espacios en RS232

Cuando se hace uso de un canal «t» para listar programas, se imprime un doble espacio entre las palabras claves como, por ejemplo, es el caso de PRINT PAPER o THEN PRINT. El error se debe a la imposibilidad de trabajar con el bit 0 de los señalizadores («flags»).

8. El problema de CLOSE

Si se hace **Break** en medio de una sentencia CLOSE # que haga referencia a un canal del Interface, la memoria que ese canal utilice no se libera. De esta forma se desperdician grandes cantidades de memoria. No es fácil contrarrestar los efectos de este error. La liberación de ese espacio de memoria no se conseguirá haciendo CLEAR #. Es necesario realizar un NEW. En la nueva ROM se ha corregido este problema añadiendo en la posición X1741H la instrucción SET 7,(HL).

9. Problemas al utilizar un valor demasiado bajo para RAMTOP

Si se intenta ejecutar un SAVE* «m» o LOAD* «m», con un valor de RAMTOP demasiado bajo, se puede afectar fatalmente a la máquina dejando, además, el motor en marcha. Se ha corregido este problema en la nueva ROM mejorando las comprobaciones de espacio libre en memoria.

10. Problemas con la red

Si se hace SAVE* N y, por casualidad, el cuarto byte de la memoria intermedia (buffer) «N» final es 205, se produce un error fatal para el sistema. Está provocado por la alteración del contenido del registro IX. Afortunadamente, las posibilidades de que esto ocurra son muy pequeñas.

BIBLIOGRAFIA

«The Working Spectrum», por David Lawrence. Sunshine Books.
«Programming the Z80», por Rodney Zaks. Sybex Inc.
«ZX Spectrum BASIC Programming», por Steven Vickers y Robin Bradbeer. Sinclair Research Ltd.
«ZX Microdrive and Interface 1 manual». Sinclair Research Ltd.
«The Complete Spectrum ROM Disassembly», por Dr. Ian Logan y Dr. Frank O'Hara. Melbourne House Ltd.



GUÍA PRÁCTICA DE BASIC del ZX-81 y del SPECTRUM

por **RAMÓN ROVIRA SOLIGÓ**

Libro n.° 172

148 págs.
31 figuras

SUMARIO

Introducción a los ordenadores - Instalación y mantenimiento - La familiarización con el ZX-81 - Empezando a programar - Instrucciones principales - Operaciones y funciones - El juego de caracteres - Las cadenas - Condiciones y operadores lógicos - Programación con bucles - Conjuntos, vectores y matrices - Gráficos - La impresora - El almacenamiento de datos en cassette - Estructura de la memoria - Acceso a la memoria - PEEK y POKE - Variables del sistema - Instrucción a código máquina - Ahorro de memoria - Comparación con otros BASICs - Códigos de error - El ZX-Spectrum - Teclado - Características adicionales del ZX-Spectrum sobre el ZX-81 - Color; sonido; separación de instrucciones en una misma línea - Instrucciones y comandos BASIC - Gráficos de alta resolución - Almacenamiento de programas y datos en cassette - Juego de caracteres - Mensajes de error - Microdrives - Mapa de memoria

ACCESO RÁPIDO AL VIC-20

por **TIM HARTNELL**

Libro n.° 173

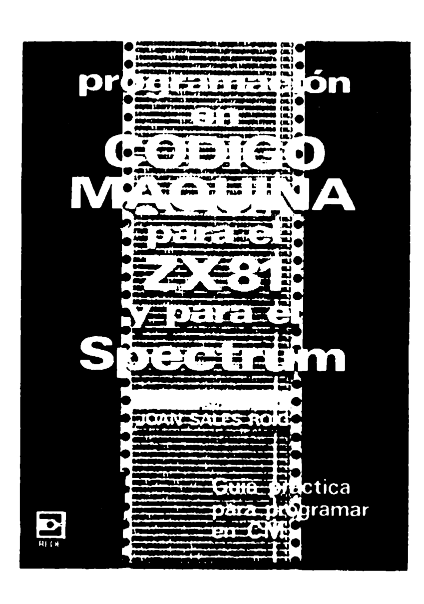
148 págs.

LISTADOS EN ESPAÑOL DE TODOS LOS PROGRAMAS



SUMARIO

Números aleatorios - Mecanismos simples de toma de decisión - La ruleta rusa - El "núcleo de programa" , la premisa para buenos juegos - Realización de gráficos - Cómo se forma la biblioteca de programas propia - For/Next - Cara o cruz - Cálculos numéricos - Ideas generales sobre la confección de programas - Percepción extrasensorial - Mata al asesino - ¿Cuánto tiempo me queda? - Jugando con las musas - Búsqueda de coordenadas - Tragaperras - Perdidos en el espacio - Vectores ("arrays") - La araña dimensional - "Pesky piksy" - Alienígenas y asteroides - Funciones gráficas en color - Serpientes y escaleras - Carreras de coches - Juego de cartas - Acertijo - El juego de las damas - Primeros pasos hacia las estrellas - Frenesí - Posibilidades musicales del VIC-20 - Instrucciones READ y DATA - Maestro - Escalas de dificultad - La cámara de ecos - Qué más puede hacer con su VIC-20 - El rey de las cavernas - Alfa - Cambio de papeles - El colgado (las nueve vidas) - Nim - El juego de la vida - Cubik - ¿Cuál es el número elegido? - Lógica - "Blackjack" - Kalki, el adivinador del pensamiento - "Chemin de fer" - "Craps" - "Esperanza de vida" - "La pajarera" - El séptimo cielo - Alunizaje - Laberinto - El profesor - La conversión de temperaturas - El acertijo de la multiplicación - Cuadrados - Apéndices: Comandos; Instrucciones; Funciones; Código de pantalla y A$



PROGRAMACIÓN EN CÓDIGO MÁQUINA PARA EL ZX-81 Y PARA EL SPECTRUM

por **JOAN SALES ROIG**

Libro n.° 175

158 págs.
10 figuras

SUMARIO

Introducción - Decimal, Hexadecimal, Binario - BASIC y Código Máquina - Los registros - Código Máquina en el ZX-81 - La instrucción LD - Aritmética sencilla: ADD, SUB, INC, DEC - Los señalizadores (flags) - Instrucciones relacionadas con flags - Saltos: absolutos, relativos y condicionales - La pila ("stack") - Trabajando con bits - Bucles y subrutinas - Instrucciones con repetición - Juego completo de instrucciones del Z-80 - Estructura de la pantalla: D- ILE - El teclado - Conexiones MIC y EAR - Técnicas de programación - Aplicación práctica: juego "Comecocos" - Direcciones útiles de la ROM - Cálculos más complejos con la ROM - Instrucciones del Z-80 - Tabla de alteraciones de flags - Tablas de saltos relativos hacia adelante y hacia atrás - Mapa de memoria - La estructura de pantalla en el ZX Spectrum - El teclado en el ZX Spectrum - El altavoz y las conexiones MIC y EAR en el Spectrum - Consideraciones para aplicar el contenido del libro al ZX Spectrum - Modificaciones en el juego "Comecocos"

ediciones técnicas



BARCELONA
(ESPAÑA)